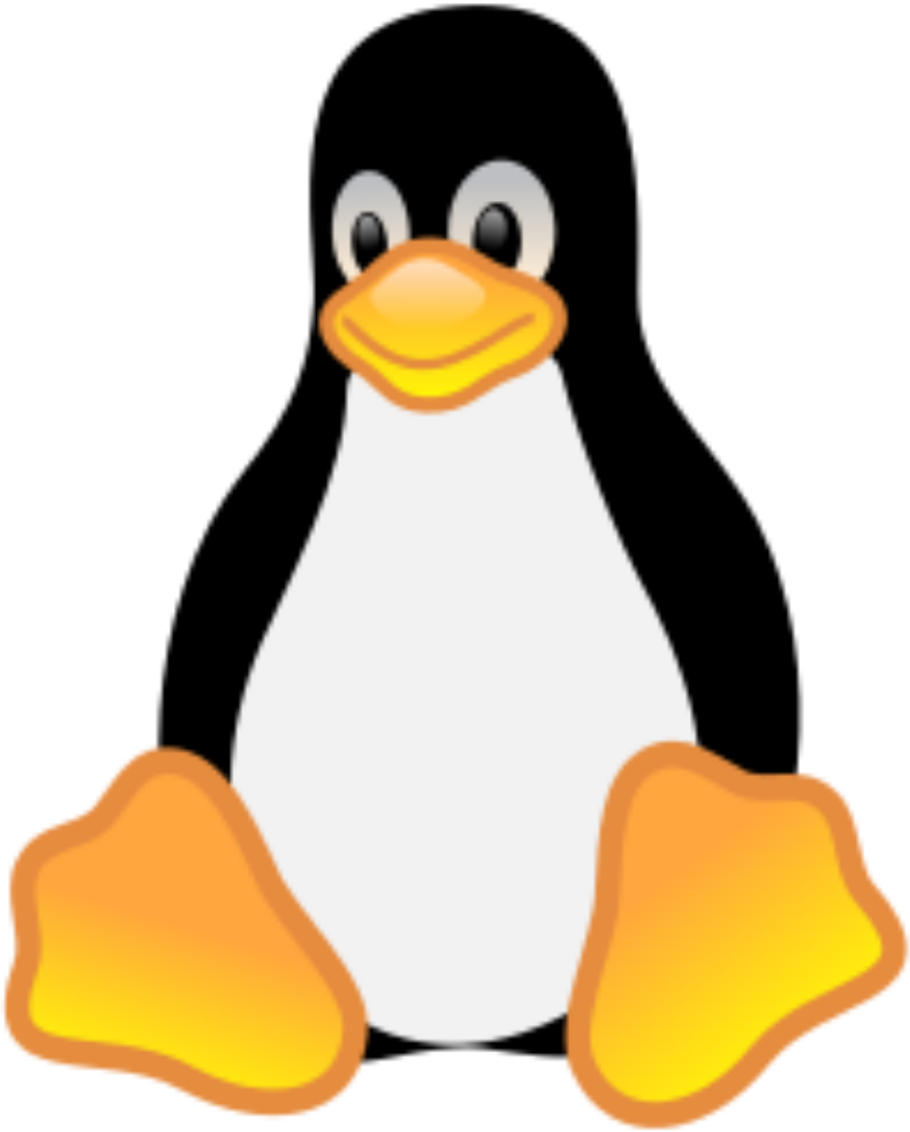
Kendi Linux'unu Yap

Temel Sistem



Yazar

- Bayram KARAHAN
- Celalettin AKARSU

Paket Derleme

- Bayram KARAHAN
- Celalettin AKARSU

İletişim

- https://github.com/kendilinuxunuyap
- https://kendilinuxunuyap.github.io
- kendilinuxunuyap@gmail.com

ISBN: 978-625-00-6004-9

Yayın Yılı: Nisan 2026

Ön Söz

Bu kitap, açık kaynak ve özgür yazılım dünyasına ilgi duyan herkes için hazırlanmıştır. Amacı, Türkçe kaynak eksikliğini gidermek ve kendi Linux dağıtımını oluşturmak isteyenlere yol göstermektir.

İçerdiği adımları dikkatle takip ederek, sıfırdan bir sistem derleyebilir ve son kullanıcıların kolayca kurup kullanabileceği bir ISO dosyası haline getirebilirsiniz. Bu rehber, basit düzeyde bir dağıtım inşa etmekten kurulabilir bir medya (ISO) hazırlamaya kadar tüm süreci adım adım açıklamaktadır.

Açık kaynak bir sistem, kullanıcıya yalnızca bir işletim sistemi sunmakla kalmaz; aynı zamanda yazılım bağımsızlığının kapılarını aralar. Kapalı kaynak sistemlerde kullanıcılar, üretici firmaların belirlediği sınırlar içinde hareket etmek zorundadır. Bu durum, hem güvenlik hem de esneklik açısından ciddi kısıtlamalar doğurur. Oysa açık kaynak dünyasında, kullanılan her bileşenin kaynağına erişebilir, üzerinde değişiklik yapabilir ve sistemi kendi ihtiyaçlarınıza göre uyarlayabilirsiniz. Bu özgürlük, özellikle kurumlar ve geliştiriciler için bağımsız bir ekosistem yaratmanın temel taşlarından biridir.

Kendi dağıtımınızı geliştirmek ise bu bağımsızlığı bir adım öteye taşır. Kullandığınız her paket, yapılandırma ve araç zinciri tamamen sizin kontrolünüzdedir. Bu sayede dışa bağımlılığı azaltabilir, sisteminizin işlevlerini kendi güvenlik ve performans kriterlerinize göre şekillendirebilirsiniz. Ayrıca, yerelleştirme ve özgün ihtiyaçlara özel optimizasyonlar yaparak sadece bir kullanıcı değil, aynı zamanda kendi yazılım ekosisteminizin yöneticisi haline gelirsiniz. Açık kaynak bu anlamda yalnızca bir tercih değil, uzun vadede teknolojik egemenliğin anahtarıdır.

Bu çalışmanın hazırlanma sürecinde değerli bilgi birikimi, sabrı ve rehberliğiyle büyük katkılarda bulunan **Ali Rıza KESKİN**'e özel bir teşekkür borçluyum. Belgenin teknik içeriğinin olgunlaşmasında, yapısal düzenlemelerinde ve özellikle açık kaynak ekosistemine dair derinlemesine görüşleriyle yön vermesinde önemli bir rol oynamıştır. Katkıları, yalnızca bu dokümanın niteliğini artırmakla kalmamış, aynı zamanda açık kaynak topluluğuna yönelik paylaşım kültürünün de güzel bir örneğini oluşturmuştur. Kendisine emekleri, desteği ve ilham verici katkıları için içtenlikle teşekkür ederim.

İçindekiler

Temel Kavramlar

Bu dokümanda, bir cihaz açıldığında kendi kendine kurulabilen ve kullanıcıyla iletişim kurabilen bir sistemin nasıl yapılacağı anlatılmaktadır.

Linux Nedir?

Linux, Linus Torvalds tarafından geliştirilen açık kaynaklı bir çekirdektir. Tek başına tam bir işletim sistemi değildir. Genellikle GNU araçları ile birlikte kullanılır. Bu yüzden çoğu zaman GNU/Linux olarak adlandırılır.

GPL Nedir?

GPL, yazılımların özgürce kullanılmasını, değiştirilmesini ve paylaşılmasını sağlayan bir lisans türüdür. Bu lisans sayesinde kaynak kodları herkes tarafından görülebilir.

GNU Nedir?

GNU, özgür yazılım geliştirmek amacıyla başlatılmış bir projedir. Linux çekirdeği ile birlikte kullanıldığında tam bir işletim sistemi ortaya çıkar.

Açık Kaynak Nedir?

Açık kaynak, bir yazılımın kaynak kodlarının herkese açık olması demektir. İsteyen kişiler bu kodları inceleyebilir ve geliştirebilir.

Dağıtım Nedir?

Dağıtım, Linux çekirdeği ve çeşitli yazılımların bir araya gelmesiyle oluşan hazır işletim sistemidir. Farklı dağıtımlar farklı amaçlar için hazırlanır.

Yazılım Bağımsızlığı

Açık kaynak yazılımlar sayesinde kullanıcılar belirli bir firmaya bağlı kalmaz. Bu da daha özgür ve esnek kullanım sağlar.

Temel Linux Sistemi Oluşturma

Bu bölümde **Linux çekirdeği** kullanarak çalışan temel bir Linux sistemi oluşturacağız. Amacımız, dağıtım geliştirme sürecinde bir başlangıç noktası hazırlamak. Sistemi hazırlamak için **kernel, initrd.img, grub.cfg** olması yeterlidir.

Bu sistemi hazırlarken debian üzerinde kurulu paketlere ihtiyacımız var. Bunlar;

- **busybox-static:**
  BusyBox temel linux komutlarını içerisinde barındıran bir uygulamadır.
- **grub-mkrescue:**
  Bir dizini ISO dosyasına dönüştüren araç.
- **qemu-system-x86:**
  ISO dosyasını test edebileceğimiz sanal bir makine uygulaması.

Debian tabanlı bir sistemde bu uygulamaları şu komutla yükleyebilirsiniz:

```
# Paketleri kuralım
sudo apt install busybox-static qemu-system-x86 qemu-system-gui grub-pc-bin grub-efi-amd64-bin grub-common xorriso mtools -y
```

Adım Adım Hazırlama

1. Çalışma Dizini Hazırlama

İlk olarak sistem dosyalarını barındıracak bir dizin oluşturalım:

```
mkdir -p $HOME/temellinux/rootfs
cd $HOME/temellinux/rootfs
```

2. BusyBox'u Kopyalama

BusyBox ikili dosyasını kök dosya sistemine kopyalayın:

```
cp /bin/busybox ./busybox
```

3. `init` Betiğini Oluşturma

Sistem açıldığında kernel'in çalıştıracağı init betiğini oluşturun:

```
nano init
```

İçerik:

```
#!/busybox sh
PATH=/bin
/busybox --install -s /bin
exec /bin/sh
```

Dosyayı kaydedip çıkın, ardından çalıştırılabilir hale getirin:

```
chmod +x init
```

4. initrd.img Dosyasını Paketleme

Tüm kök dosya sistemi içeriğini initramfs imajına dönüştürün:

```
find . | cpio -H newc -o > ../initrd.img
gzip -9 ../initrd.img
```

5. GRUB Yapılandırmasını Hazırlama

GRUB'un boot edebilmesi için gerekli dosya yapısını oluşturun:

```
mkdir -p $HOME/temellinux/iso/boot/grub
nano $HOME/temellinux/iso/boot/grub/grub.cfg
```

grub.cfg İçerik:

```
set timeout=5
set default=0

menuentry "Temel Linux" {
    linux /boot/vmlinuz
    initrd /boot/initrd.img
}
```

6. Kernel ve initrd'yi iso dizinine kopyalayın:

```
cp /boot/vmlinuz-$(uname -r) $HOME/temellinux/iso/boot/vmlinuz
cp $HOME/temellinux/initrd.img $HOME/temellinux/iso/boot/initrd.img
```

Bu aşamaya geldiyseniz aşağıdaki gibi bir dizin yapısı oluşmuş olmalı:

```
$HOME/temellinux/iso/boot/grub/grub.cfg
$HOME/temellinux/iso/boot/initrd.img
$HOME/temellinux/iso/boot/vmlinuz
```

7. ISO Dosyası Oluşturma

Boot edilebilir ISO dosyasını oluşturun:

```
grub-mkrescue -o $HOME/temellinux/temellinux.iso $HOME/temellinux/iso
```

8. Sistemi Test Etme

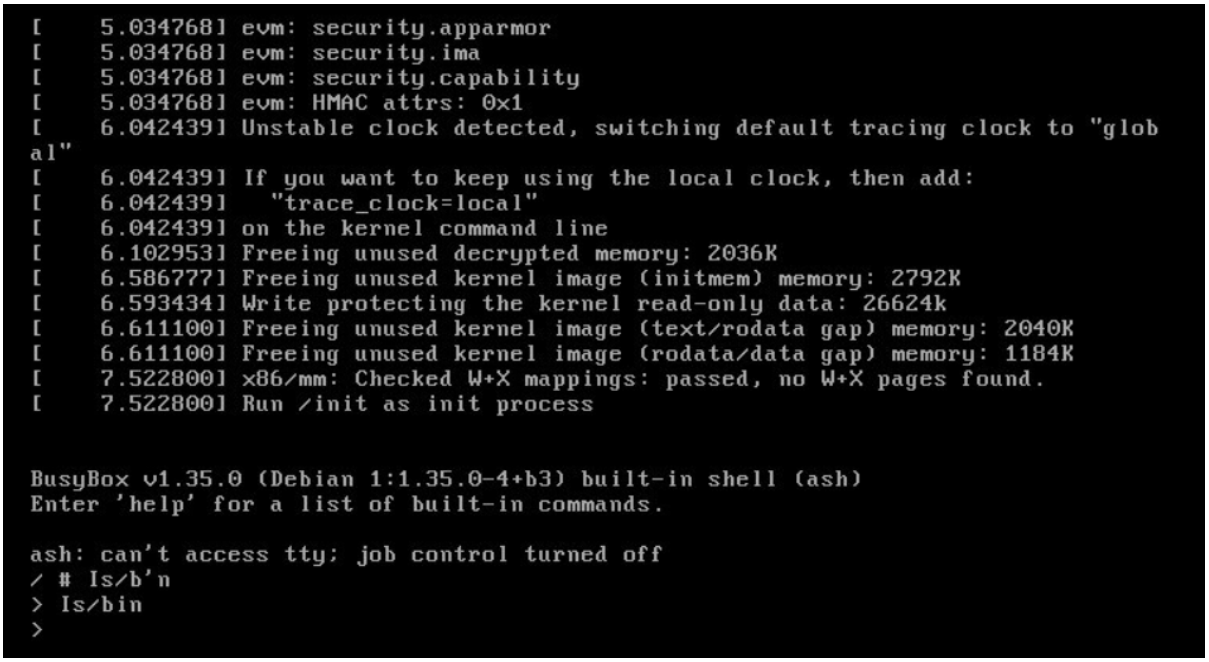
QEMU ile sistemi test edin:

```
qemu-system-x86_64 -cdrom $HOME/temellinux/temellinux.iso -m 1024M
```

Başarılı bir şekilde açılırsa, BusyBox'un sağladığı temel komutları çalıştırabilirsiniz.

Sonuç

Bu adımlar sonunda, yalnızca **BusyBox** ve **Kernel** ile çalışan basit bir Linux sistemi elde etmiş olacaksınız. Bu temel sistem, daha gelişmiş bir dağıtım hazırlamak için rehberlik edecektir.



Kaynaklar:

- https://www.subrat.info/build-kernel-and-userspace/ - 08/07/2025
- https://medium.com/@chienhaotan/compiling-and-running-a-minimal-kernel-with-busybox-bfc45a991017 - 08/07/2025
- https://wiki.archlinux.org/title/GRUB_(T%C3%BCrk%C3%A7e) - 08/07/2025
- https://gist.github.com/bluedragon1221/a58b0e1ed4492b44aa530f4db0ffef85 - 09/07/2025

GNU Araçlarıyla Kendi Linux Dağıtımını Hazırlamak

Bir önceki bölümde, temel bir sistem hazırlamayı adım adım öğrenmiştik. O çalışmada, temel işlevselliğe sahip, kendi kendine açılabilen, küçük ama işlevsel bir Linux ortamı kurduk. Şimdi ise bu yolculuğu bir adım daha ileri taşıyoruz!

Artık daha güçlü, daha esnek ve daha geniş özelliklere sahip bir sistem inşa etmenin zamanı geldi. Bu bölümde, **GNU araçlarıyla** nasıl bir dağıtım hazırlayabileceğimizi detaylı şekilde anlatacağız. Peki neden busybox'la yetinmeyip GNU araçlarına geçiyoruz? İşte tam da burada kararımızın nedenlerini birlikte keşfedelim.

BusyBox:

BusyBox, bize temel sistem oluşturmada büyük kolaylık sağladı. Tek bir binary ile onlarca temel komutu çalıştırabildik, disk alanından tasarruf ettik ve bağımlılıklarla uğraşmadık. Busybox özellikle gömülü sistemlerde ve kurtarma ortamlarında adeta bir hayat kurtarıcı. **BusyBox** kurulumu ve kullanımıyla ilgili detayları **Yardımcı Konular** bölümünde bulabilirsiniz.

Avantajlar

- **Küçük boyut**, **Taşınabilirlik** : Tek binary ile onlarca komut.
- **Düşük bağımlılık**: Ekstra kütüphane gerektirmez.
- **Kolay kurulum**: Derlemesi ve kurulumu oldukça basit.
- **Çok amaçlı kullanım**: Tek dosyada tüm temel araçlar.

Dezavantajlar

- **Sınırlı özellikler**: Gelişmiş parametre seçenekleri yok.
- **Standart dışı davranışlar**, **Script uyumsuzluğu**: GNU araçlarıyla tam uyumlu değil. Karmaşık bash scriptleri çalışmayabilir.
- **Performans sınırlamaları**: Büyük veri işlemlerinde yetersiz.
- **Hata ayıklama zorlukları**: Detaylı debug için yetersiz.

İşte bu nedenle, artık GNU araçlarıyla donatılmış daha güçlü bir sistem hazırlama zamanı geldi.

Yeni Hedef: Tam Teşekküllü Bir Linux Dağıtımı

Bu yolculukta, aşağıdaki yeteneklere sahip bir sistem kurmayı hedefliyoruz:

- Temel **Linux komutlarını** eksiksiz şekilde çalıştırabilmek
- **Bash kabuğu** ile tam işlevli bir TTY ortamı sunmak
- Farklı **sıkıştırma formatlarıyla** dosya işlemleri yapmak
- **Kernel modüllerini** yönetebilmek
- **Initrd** (Initial RAM Disk) oluşturabilmek
- **GRUB** ile sistem önyüklemesi yapabilmek
- Sistemi hem **kurulabilir** hem de **live (canlı)** modda çalıştırabilmek
- **İnternete bağlanabilir** olmak
- **SSH** ile uzaktan yönetim imkanı sağlamak
- Metin düzenlemek için bir **editör (nano)** sunmak

Kurulum Dizini: Dağıtımın Oluşacağı Yer

Dağıtım sürecinde tüm çalışma dosyalarımızı saklayacağımız bir hedef dizin belirlemeliyiz.

Bu doküman boyunca varsayılan çalışma dizini olarak şunu kullanacağız:

`$HOME/distro/rootfs`

Burada **$HOME** ne demek? Bu, Linux ortamında aktif olan kullanıcının ev dizinini ifade eden standart bir değişkendir. Örneğin, sistemde giriş yapan kullanıcı adınız **uzman** ise; **$HOME = /home/uzman** konumunu belirtir.Bu sayede doküman boyunca paylaşılan tüm komutlarda **$HOME** ifadesi sizin kendi kullanıcı dizininize göre otomatik olarak uyarlanacak.

Artık ortam hazır!

Ortamın hazırlanmasından sonra bazı konuları bilmemiz gerekmektedir. Bunlar;

1. Derleme(Dinamik/Static)
2. BusyBox
3. chroot Kullanımı
4. Kernel/Modül Derleme
5. İso Oluşturma
6. Canlı Sistem Oluşturma Kullanma
7. qemu Kullanmı
8. cfdisk Kullanımı
9. Canlı Sistemden Kurulum Yapma

Burada liste halinde verilen konu başlıkları bu dokümanın **Yardımcı Konular** bölümünde anlatılmaktadır.

Bundan sonraki adımlarda kendi dağıtımımızı şekillendirmeye başlayacağız!

İşte Tüm Bu Yapı İçin İhtiyacımız Olan Paketler

0- base-file	25- elfutils	50- popt
1- glibc	26- libselinux	51- icu
2- readline	27- tar	52- iproute2
3- ncurses	28- zlib	53- net-tools
4- bash	29- brotli	54- dhcp
5- openssl	30- curl	55- openrc
6- acl	31- shadow	56- rsync
7- attr	32- file	57- kbd
8- libcap	33- eudev	58- kernel
9- libpcre2	34- cpio	59- dialog
10- gmp	35- libsepol	60- live-boot
11- coreutils	36- kmod	61- live-config
12- util-linux	37- audit	62- parted
13- grep	38- libxcrypt	63- busybox
14- sed	39- libnsl	64- nano
15- mpfr	40- libbsd	65- grub
16- gawk	41- libtirpc	66- efibootmgr
17- findutils	42- e2fsprogs	67- efivar
18- gcc	43- dosfstools	68- libssh
19- libcap-ng	44- initramfs-tools	69- openssh
20- sqlite	45- libxml2	70- pam
21- gzip	46- expat	71-
22- xz-utils	47- libmd	72-
23- zstd	48- libaio	73-
24- bzip2	49- lvm2	74-

Bağımlılıklar Zinciri: Doğru Sıralama Şart!

Linux paketinin sorunsuz çalışabilmesi için **bağımlı olduğu tüm paketlerin önceden derlenmiş olması gerekir**. Örneğin; **bash** paketini çalıştırmak için: **readline** ve **ncurses** paketleri gereklidir. **readline** ve **ncurses** için ise **glibc** paketi gereklidir. Bu sayfadaki liste bağımlılıklarına göre sıralandı. Liste sırasına göre ilerlemek oldukça önemlidir. Her paketin derleme aşaması, ayrı başlıklar altında detaylı bir şekilde anlatılacaktır.

Hazırlık: Gerekli Derleme Araçlarını Kurun!

Paket derleme işlemine başlamadan önce, aşağıdaki temel araçları sisteminize kurmalısınız.

```
sudo apt update
sudo apt-get install debootstrap xorriso mtools make squashfs-tools gcc wget unzip xz-utils tar zstd fakeroot \
autoconf automake autotools-dev make meson cmake ninja-build pkgconf patch libtool grub-pc grub-pc-bin
```

base-file

Linux sistemimiz için öncelikle temel ayarlamalar ve gerekli dosya-dizin yapıları oluşturulmalıdır. Bu yapıyı hazırladıktan sonra, diğer tüm paketleri ve sistemi bu temel üzerine inşa edeceğiz.

Aslında bir Linux sisteminin en temel paketi **glibc**'dir. Ancak, **glibc** derlenip yüklenmeden önce, sistem için gerekli temel dizin yapısının hazır olması gerekir. Bu nedenle **base-file** adında özel bir paket oluşturduk.

base-file Komutları

Aşağıdaki komutlarla temel dosya ve dizin yapısını hazırlıyoruz:

```
display=":$(ls /tmp/.X11-unix/* | sed 's#/tmp/.X11-unix/X##' | head -n 1)"      # Açık ekran tespiti
user=$(who | grep '('$display')' | awk '{print $1}')    # Ekranı açan kullanıcı tespiti
mkdir -p /home/$user/distro/build       # Derleme dizini yoksa oluşturuluyor
mkdir -p /home/$user/distro/rootfs      # Sistemin oluşturulacağı dizin yoksa oluşturuluyor
rm -rf /home/$user/distro/build/*       # İçeriği temizleniyor
cp -prfv files/* /home/$user/distro/build/      # Ek dosyalar kopyalanıyor
cd /home/$user/distro/build     # Derleme dizinine geçiliyor

mkdir -p bin dev etc home lib64 proc root run sbin sys usr var etc/kly tmp tmp/kly/kur \
var/log var/tmp usr/lib64/x86_64-linux-gnu usr/lib64/pkgconfig \
usr/local/{bin,etc,games,include,lib,sbin,share,src}
ln -s lib64 lib
cd var && ln -s ../run run && cd -
cd usr && ln -s lib64 lib && cd -
cd usr/lib64/x86_64-linux-gnu && ln -s ../pkgconfig pkgconfig && cd -

bash -c "echo -e \"/bin/sh \n/bin/bash \n/bin/rbash \n/bin/dash\" >> /home/$user/distro/build/etc/shell"
bash -c "echo 'tmpfs /tmp tmpfs rw,nodev,nosuid 0 0' >> /home/$user/distro/build/etc/fstab"
bash -c "echo '127.0.0.1 kly' >> /home/$user/distro/build/etc/hosts"
bash -c "echo 'kly' > /home/$user/distro/build/etc/hostname"
bash -c "echo 'nameserver 8.8.8.8' > /home/$user/distro/build/etc/resolv.conf"
echo root:x:0:0:root:/root:/bin/sh > /home/$user/distro/build/etc/passwd
chmod 755 /home/$user/distro/build/etc/passwd
cp -prfv /home/$user/distro/build/* /home/$user/distro/rootfs/
```

Yukarıdaki işlemleri daha düzenli ve fonksiyonel hale getirmek için aşağıdaki şablon script yapısını kullanacağız.

Şablon Script Yapısı

Aşağıda, tüm paketlerde kullanılacak şablon script yapısı örneği verilmiştir:

```
#!/usr/bin/env bash
version=""
name=""
depends=""
source=""
display=":$(ls /tmp/.X11-unix/* | sed 's#/tmp/.X11-unix/X##' | head -n 1)"      # Açık ekran tespiti
user=$(who | grep '('$display')' | awk '{print $1}')    # Kullanıcı tespiti
ROOTBUILDDIR="/home/$user/distro/build"         # Derleme konumu
BUILDDIR="/home/$user/distro/build/build-${name}-${version}"    # Paket derleme dizini
DESTDIR="/home/$user/distro/rootfs"     # Yüklenecek dizin
PACKAGEDIR=$(pwd)       # Derleme talimatının bulunduğu dizin
SOURCEDIR="/home/$user/distro/build/${name}-${version}"         # Kaynak dizini

initsetup(){
        mkdir -p $ROOTBUILDDIR
        rm -rf $ROOTBUILDDIR/*
        cd $ROOTBUILDDIR
        wget ${source}
        dowloadfile=$(ls | head -1)
        filetype=$(file -b --extension $dowloadfile | cut -d'/' -f1)
        if [ "${filetype}" == "???" ]; then unzip ${dowloadfile}; else tar -xvf ${dowloadfile}; fi
        director=$(find ./* -maxdepth 0 -type d)
        directorname=$(basename ${director})
        if [ "${directorname}" != "${name}-${version}" ]; then mv $directorname ${name}-${version}; fi
        mkdir -p $BUILDDIR && mkdir -p $DESTDIR && cd $BUILDDIR
}

setup(){        # Derleme öncesi kaynak dosyaların hazırlanması }
build(){        # Paketin derlenmesi    }
package(){      # Derlenen dosyaların yüklenmesi        }

initsetup
setup
build
package
```

Şablonda Kullanılan Sabit Bilgiler

Şablon script içinde geçen bazı sabit değişkenler şunlardır:

- **ROOTBUILDDIR:** /home/$user/distro/build → Derleme dizini
- **BUILDDIR:** /home/$user/distro/build/build-${name}-${version} → Paket derleme dizini
- **DESTDIR:** /home/$user/distro/rootfs → Yükleme dizini
- **PACKAGEDIR:** $(pwd) → Derleme scriptinin bulunduğu dizin
- **SOURCEDIR:** /home/$user/distro/build/${name}-${version} → Kaynak dizin

Bu değişkenler sayesinde uzun dizin yolları yerine sadece kısaltmaları kullanıyoruz. Bu aslında bir çeşit takma ad (alias) kullanımına benzer. Örneğin, kaynak dizinde işlem yapmak için sadece **$SOURCEDIR** kullanmanız yeterlidir. Bu yapılar tüm paketlerde geçerli olacak.

base-file scriptine benzer şekilde, diğer paketler için de aynı şablon yapısı kullanılacaktır. Böylece her paket için aynı temel adımları tekrar tekrar yazmanıza gerek kalmayacak. Ayrıca, hata olması durumunda ilgili fonksiyon üzerinden doğrudan müdahale ederek kolayca çözüm sağlayabileceksiniz.

Şablon Scripte Uygun base-file Scripti

Aşağıda **base-file** paketine ait tam script örneği verilmiştir:

```
#!/usr/bin/env bash
version="1.0"
name="base-file"
depends=""
description="sistemin temel yapısı"
source=""

display=":$(ls /tmp/.X11-unix/* | sed 's#/tmp/.X11-unix/X##' | head -n 1)"
user=$(who | grep '('$display')' | awk '{print $1}')
ROOTBUILDDIR="/home/$user/distro/build"
BUILDDIR="/home/$user/distro/build/build-${name}-${version}"
DESTDIR="/home/$user/distro/rootfs"
PACKAGEDIR=$(pwd)
SOURCEDIR="/home/$user/distro/build/${name}-${version}"
initsetup(){
        mkdir -p $ROOTBUILDDIR
        rm -rf $ROOTBUILDDIR/*
        cd $ROOTBUILDDIR
        mkdir -p $BUILDDIR && mkdir -p $DESTDIR && cd $BUILDDIR
}
setup(){
        cp -prfv $PACKAGEDIR/files/* $BUILDDIR/
}
build(){
        echo ""
}
package(){
        mkdir -p bin dev etc home lib64 proc root run sbin sys usr var etc/kly \
        tmp tmp/kly/kur var/log var/tmp usr/lib64/x86_64-linux-gnu \
        usr/lib64/pkgconfig usr/local/{bin,etc,games,include,lib,sbin,share,src}
        ln -s lib64 lib
        cd var && ln -s ../run run && cd -
        cd usr && ln -s lib64 lib && cd -
        cd usr/lib64/x86_64-linux-gnu && ln -s ../pkgconfig pkgconfig && cd -
        printf "/bin/sh\n/bin/bash\n/bin/rbash\n/bin/dash\n" >> "$BUILDDIR/etc/shell"
        echo 'tmpfs /tmp tmpfs rw,nodev,nosuid 0 0' >> "$BUILDDIR/etc/fstab"
        echo '127.0.0.1 kly' >> "$BUILDDIR/etc/hosts"
        echo 'kly' > "$BUILDDIR/etc/hostname"
        echo 'nameserver 8.8.8.8' > "$BUILDDIR/etc/resolv.conf"
        echo 'root:x:0:0:root:/root:/bin/sh' > "$BUILDDIR/etc/passwd"
        chmod 755 $BUILDDIR/etc/passwd
        cp -prfv $BUILDDIR/* $DESTDIR/
}
initsetup
setup
build
package
```

Not: Yukarıdaki scriptin sorunsuz çalışabilmesi için ek dosyaları indirin. İndirdiğiniz **files.tar** dosyasını istediğiniz bir konumda **base-file** adında bir dizin oluşturarak oraya açınız. **base-file** dizini içine yukarıdaki script kodlarını build adlı bir dosya oluşturup yapıştırın ve kaydedin. **base-file** dizininde bir terminal açarak aşağıdaki komutları sırasıyla çalıştırın.

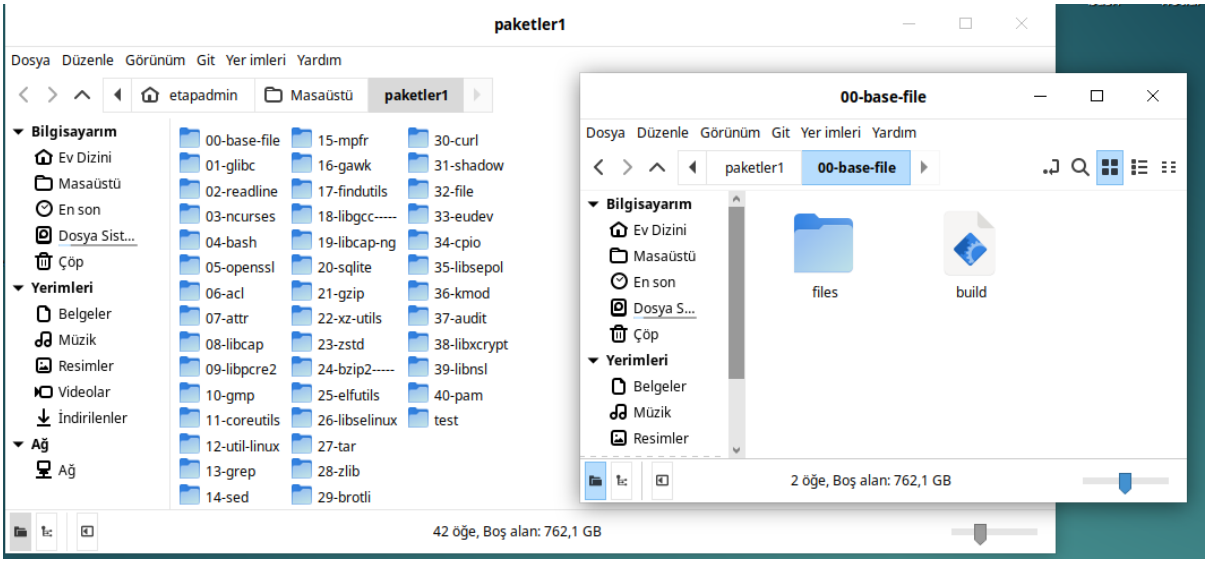
```
chmod 755 build
sudo ./build
```

Paket Derleme Yöntemi

base-file paketi ilk paket olduğu için detaylı anlatıldı. Bundan sonraki paketlerde sadece **şablon bir script dosyası** verilecek. Eğer paketin ek dosyaları varsa, bunlar **files.tar** şeklinde sunulacak.

Her paket için istediğiniz bir konumda yeni bir dizin oluşturun. Varsa, ilgili **files.tar** arşivini dizin içinde açın.

Aşağıda, test amaçlı derlediğim bazı paketler ile **base-file** için oluşturulan dizin yapısı görülmektedir:



Derleme Scripti Kullanımı

Her paket için **build** adında bir dosya oluşturup, script kodlarını içine yapıştırın ve kaydedin.

Sonrasında, o dizin içinde terminal açarak aşağıdaki komutları çalıştırabilirsiniz:

```
chmod 755 build
sudo ./build
```

glibc

glibc linux dağıtımlarında bütün uygulamaların çalışmasını sağlayan en temel C kütüphanesidir. **glibc** temel kütüphane olduğu için ilk bu paketi derleyeceğiz.

glibc Script Dosyası

Debian ortamında bu paketin derlenmesi için; **sudo apt install make bison gawk diffutils gcc gettext grep perl sed texinfo libtool** komutuyla paketin kurulması gerekmektedir.

```
#-------------------------------------------------------------------------------------------------------------------
#!/usr/bin/env bash
version="2.39"
name="glibc"
description="temel kütüphane"
source="https://ftp.gnu.org/gnu/libc/${name}-${version}.tar.gz"
export CC="gcc"
export CXX="g++"
display=":$(ls /tmp/.X11-unix/* | sed 's#/tmp/.X11-unix/X##' | head -n 1)"
user=$(who | grep '('$display')' | awk '{print $1}')  # Detect the user using such display
ROOTBUILDDIR="/home/$user/distro/build" # Derleme konumu
BUILDDIR="/home/$user/distro/build/build-${name}-${version}" #Derleme yapılan paketin derleme konumu
DESTDIR="/home/$user/distro/rootfs" #Paketin yükleneceği sistem konumu
PACKAGEDIR=$(pwd) #paketin derleme talimatının verildiği konum
SOURCEDIR="/home/$user/distro/build/${name}-${version}" #Paketin kaynak kodlarının olduğu konum
initsetup(){
mkdir -p  $ROOTBUILDDIR #derleme dizini yoksa oluşturuluyor
rm -rf $ROOTBUILDDIR/* #içeriği temizleniyor
cd $ROOTBUILDDIR #dizinine geçiyoruz
wget ${source}
for f in *\ *; do mv "$f" "${f// /}"; done #isimde boşluk varsa silme işlemi yapılıyor
dowloadfile=$(ls|head -1)
filetype=$(file -b --extension $dowloadfile|cut -d'/' -f1)
if [ "${filetype}" == "???" ]; then unzip  ${dowloadfile}; else tar -xvf ${dowloadfile};fi
director=$(find ./* -maxdepth 0 -type d)
directorname=$(basename ${director})
if [ "${directorname}" != "${name}-${version}" ]; then mv $directorname ${name}-${version};fi
mkdir -p $BUILDDIR && mkdir -p $DESTDIR && cd $BUILDDIR
}
setup() {
cp -prvf $PACKAGEDIR/files $BUILDDIR/
echo "slibdir=/lib64" >> configparms
echo "rtlddir=/lib64" >> configparms
$SOURCEDIR/configure --prefix=/usr --mandir=/usr/share/man --infodir=/usr/share/info --enable-bind-now \
--enable-multi-arch --enable-stack-protector=strong --enable-stackguard-randomization --disable-crypt \
--disable-profile --disable-werror --enable-static-pie --enable-static-nss --disable-nscd \
--host=x86_64-pc-linux-gnu --libdir=/lib64 --libexecdir=/lib64/glibc
}
build(){
make $DESTDIR all
}
package(){
mkdir -p ${DESTDIR}/lib64
cd $DESTDIR
ln -s lib64 lib
cd $BUILDDIR
make install DESTDIR=$DESTDIR
mkdir -p ${DESTDIR}/etc/ld.so.conf.d/ ${DESTDIR}/bin
install $BUILDDIR/files/ld.so.conf ${DESTDIR}/etc/ld.so.conf
install $BUILDDIR/files/usr-support.conf ${DESTDIR}/etc/ld.so.conf.d/
install $BUILDDIR/files/x86_64-linux-gnu.conf ${DESTDIR}/etc/ld.so.conf.d/
rm -f ${DESTDIR}/etc/ld.so.cache
install $BUILDDIR/files/locale-gen ${DESTDIR}/bin/locale-gen
install $BUILDDIR/files/revdep-rebuild ${DESTDIR}/bin/revdep-rebuild
install $BUILDDIR/files/tr_TR ${DESTDIR}/usr/share/i18n/locales/tr_TR
sed -i "s|#!/bin/bash|#!/bin/sh|g" ${DESTDIR}/usr/bin/ldd       #fix ldd shebang
cd ${DESTDIR}/lib64/ && mkdir -p x86_64-linux-gnu && cd x86_64-linux-gnu
while read -rd '' file; do
ln -s $file $(basename "$file")
done < <(find "../" -maxdepth 1 -type f -iname "*" -print0)
${DESTDIR}/sbin/ldconfig -r ${DESTDIR}  # sistem guncelleniyor
}
initsetup; setup; build;package;
```

Ek dosyayı indirmek için tıklayınız. **Paket Derleme Yöntemi** konusunda anlatıldığı gibi derleme işlemini yapınız.

glibc Test Etme

glibc kütüphanemizi **$HOME/distro/rootfs** komununa yüklendi. Şimdi bu kütüphanenin çalışıp çalışmadığını test edelim. Aşağıdaki c kodumuzu derleyelim ve **$HOME/distro/rootfs** konumuna kopyalayalım. **$HOME/** (ev dizinimiz) konumuna dosyamızı oluşturup aşağıdaki kodu içine yazalım.

```
#include<stdio.h>
void main(){
puts("Merhaba Dünya");
}
```

Program Derleme

```
cd $HOME
gcc -o merhaba merhaba.c #merhaba.c dosyası derlenir.
```

Program Yükleme

Derlenen çalışabilir merhaba dosyamızı **glibc** kütüphanemizin olduğu dizine yükleyelim.

```
cp merhaba $HOME/distro/rootfs/merhaba # derlenen merhaba ikili dosyası $HOME/distro/rootfs/ konumuna kopyalandı.
```

Programı Test Etme

glibc kütüphanemizin olduğu dizin dağıtımızın ana dizini oluyor. **$HOME/distro/rootfs/** konumuna **chroot** ile erişelim.

Aşağıdaki gibi çalıştırdığımızda bir hata alacağız.

```
sudo chroot $HOME/distro/rootfs/ /merhaba
chroot: failed to run command ‘/merhaba’: No such file or directory
```

Hata Çözümü

```
# üstteki hatanın çözümü sembolik bağ oluşturmak.
cd $HOME/distro/rootfs/
ln -s lib lib64
```

#merhaba dosyamızı tekrar chroot ile çalıştıralım. Aşağıda görüldüğü gibi hatasız çalışacaktır.

```
sudo chroot $HOME/distro/rootfs/ /merhaba
Merhaba Dünya
```

Merhaba Dünya mesajını gördüğümüzde glibc kütüphanemizin ve merhaba çalışabilir dosyamızın çalıştığını anlıyoruz. Bu aşamadan sonra **Temel Paketler** listemizde bulunan paketleri kodlarından derleyerek **$HOME/distro/rootfs/** dağıtım dizinimize yüklemeliyiz.

Derlemede **glibc** kütüphanesinin derlemesine benzer bir yol izlenecektir. **glibc** temel kütüphane olması ve ilk derlediğimiz paket olduğu için detaylıca anlatılmıştır. Diğer paketlerimizde de **glibc** için paylaşılan script dosyası gibi dosyalar hazırlayıp derlenecektir.

readline

libreadline, komut satırında girdi almasını ve düzenlemesini sağlar. Debian ortamında bu paketin derlenmesi için; **sudo apt install libreadline-dev** komutuyla paketin kurulması gerekmektedir.

```
#-----------------------------------------------------------------------------------------------------------------------
#!/usr/bin/env bash
version="8.2"
name="readline"
depends="glibc"
description="readline kütüphanesi"
source="https://ftp.gnu.org/pub/gnu/readline/${name}-${version}.tar.gz"
display=":$(ls /tmp/.X11-unix/* | sed 's#/tmp/.X11-unix/X##' | head -1)"
user=$(who | grep "(${display})" | awk '{print $1}')    # kullanıcı tespit
ROOT="/home/$user/distro"
ROOTBUILDDIR="$ROOT/build"                              # Derleme dizini
BUILDDIR="$ROOT/build/build-${name}-${version}" # Alt dizin
DESTDIR="$ROOT/rootfs"                                  # Yükleme dizini
PACKAGEDIR=$(pwd)                                       # Paket dizini
SOURCEDIR="$ROOT/build/${name}-${version}"              # Kaynak dizini
initsetup(){
        mkdir -p  $ROOTBUILDDIR #derleme dizini yoksa oluşturuluyor
        rm -rf $ROOTBUILDDIR/* #içeriği temizleniyor
        cd $ROOTBUILDDIR #dizinine geçiyoruz
        wget ${source}
        #isimde boşluk varsa silme işlemi yapılıyor
        for f in *\ *; do mv "$f" "${f// /}"; done
        dowloadfile=$(ls|head -1)
        filetype=$(file -b --extension $dowloadfile|cut -d'/' -f1)
        if [ "${filetype}" == "???" ]; then unzip  ${dowloadfile}; \
        else tar -xvf ${dowloadfile};fi
        director=$(find ./* -maxdepth 0 -type d)
        directorname=$(basename ${director})
        if [ "${directorname}" != "${name}-${version}" ]; \
        then mv $directorname ${name}-${version};fi
        mkdir -p $BUILDDIR&&mkdir -p $DESTDIR&&cd $SOURCEDIR
}
setup(){
        cp -prvf $PACKAGEDIR/files $SOURCEDIR/
        ./configure --prefix=/usr --libdir=/usr/lib64
}
build(){
        make SHLIB_LIBS="-L/tools/lib -lncursesw"
}
package(){
        make SHLIB_LIBS="-L/tools/lib -lncursesw" DESTDIR="$DESTDIR" \
         install pkgconfigdir="/usr/lib64/pkgconfig"
        install -Dm644 $SOURCEDIR/files/inputrc "$DESTDIR"/etc/inputrc
        ${DESTDIR}/sbin/ldconfig -r ${DESTDIR}           # sistem guncelleniyor
}
initsetup       # initsetup fonksiyonunu çalıştırır ve kaynak dosyayı indirir
setup           # setup fonksiyonu çalışır ve derleme öncesi kaynak dosyaların ayalanması sağlanır.
build           # build fonksiyonu çalışır ve kaynak dosyaları derlenir.
package         # package fonksiyonu çalışır, yükleme öncesi ayarlamalar yapılır ve yüklenir.
```

Ek dosyaları indirmek için tıklayınız. **Paket Derleme Yöntemi** konusunda anlatıldığı gibi derleme işlemini yapınız.

ncurses

ncurses, metin tabanlı menüler oluşturmak, renkleri ve stil ayarlamamızı sağlayan kütüphanedir. Debian ortamında bu paketin derlenmesi için; **sudo apt install libncurses-dev** komutuyla paketin kurulması gerekmektedir.

```
#-----------------------------------------------------------------------------------------------------------------------
#!/usr/bin/env bash
version="6.4"
so_ver="6"
name="ncurses"
depends="glibc"
description="ncurses kütüphanesi"
source="https://ftp.gnu.org/pub/gnu/ncurses/${name}-${version}.tar.gz"
display=":$(ls /tmp/.X11-unix/* | sed 's#/tmp/.X11-unix/X##' | head -1)"  # Display adı
user=$(who | grep "(${display})" | awk '{print $1}')                      # Display kullanıcısı

ROOT="/home/$user/distro"
ROOTBUILDDIR="$ROOT/build"                # Derleme dizini
BUILDDIR="$ROOT/build/build-${name}-${version}" # Alt dizin
DESTDIR="$ROOT/rootfs"                     # Yükleme dizini
PACKAGEDIR=$(pwd)                          # Paket dizini
SOURCEDIR="$ROOT/build/${name}-${version}" # Kaynak dizini

initsetup(){
        mkdir -p  $ROOTBUILDDIR #derleme dizini yoksa oluşturuluyor
        rm -rf $ROOTBUILDDIR/* #içeriği temizleniyor
        cd $ROOTBUILDDIR #dizinine geçiyoruz
        wget ${source}
        #isimde boşluk varsa silme işlemi yapılıyor
        for f in *\ *; do mv "$f" "${f// /}"; done
        dowloadfile=$(ls|head -1)
        filetype=$(file -b --extension $dowloadfile|cut -d'/' -f1)
        if [ "${filetype}" == "???" ]; then unzip  ${dowloadfile}; \
        else tar -xvf ${dowloadfile};fi
        director=$(find ./* -maxdepth 0 -type d)
        directorname=$(basename ${director})
        if [ "${directorname}" != "${name}-${version}" ]; \
        then mv $directorname ${name}-${version};fi
        mkdir -p $BUILDDIR&&mkdir -p $DESTDIR&&cd $SOURCEDIR
}
setup(){
        ./configure --prefix=/usr --libdir=/lib64 --with-shared  --disable-tic-depends --with-versioned-syms \
         --enable-widec --with-cxx-binding --with-cxx-shared --enable-pc-files --mandir=/usr/share/man \
         --with-manpage-format=normal --with-xterm-kbs=del --with-pkg-config-libdir=/usr/lib64/pkgconfig
}
build(){
        make
}
package(){
        make install DESTDIR=$DESTDIR
        cd $DESTDIR/lib64
        ln -s libncursesw.so.6 libtinfow.so.6
        ln -s libncursesw.so.6 libtinfo.so.6
        ln -s libncursesw.so.6 libncurses.so.6
        # Paket config dosyaları linkleniyor
        for lib in ncurses ncurses++ form panel menu; do
            if [ ! -f "$DESTDIR/usr/lib64/pkgconfig/${lib}.pc" ]; then
                ln -sv ${lib}w.pc "$DESTDIR/usr/lib64/pkgconfig/${lib}.pc"
            fi
        done
        for lib in tic tinfo tinfow ticw; do
            if [ ! -f "$DESTDIR/usr/lib64/pkgconfig/${lib}.pc" ]; then
                ln -sv ncursesw.pc "$DESTDIR/usr/lib64/pkgconfig/${lib}.pc"
            fi
        done
        # Eski sürüm desteği için sembolik linkler
        for lib in libncursesw libncurses libtinfo libpanelw libformw libmenuw; do
            ln -sv ${lib}.so.${so_ver} ${lib}.so.5
        done
        ${DESTDIR}/sbin/ldconfig -r ${DESTDIR}
}
initsetup       # initsetup fonksiyonunu çalıştırır ve kaynak dosyayı indirir
setup           # setup fonksiyonu çalışır ve derleme öncesi kaynak dosyaların ayalanması sağlanır.
build           # build fonksiyonu çalışır ve kaynak dosyaları derlenir.
package         # package fonksiyonu çalışır, yükleme öncesi ayarlamalar yapılır ve yüklenir.
```

Paket Derleme Yöntemi konusunda anlatıldığı gibi derleme işlemini yapınız.

bash

Bash, kabuk programlama dilidir. Kullanıcıların komut girmesini, program çalıştırmasını ve betikler (script) ile bir çok işlemi otomatikleştirmemizi sağlar.

```
#-----------------------------------------------------------------------------------------------------------------------
#!/usr/bin/env bash
version="5.2.21"
name="bash"
depends="glibc,readline,ncurses"
description="GNU/Linux dağıtımında ön tanımlı kabuk"
source="https://ftp.gnu.org/pub/gnu/bash/${name}-${version}.tar.gz"
# Display adı  # Display kullanıcısı
display=":$(ls /tmp/.X11-unix/* | sed 's#/tmp/.X11-unix/X##' | head -1)"
user=$(who | grep "(${display})" | awk '{print $1}')

ROOT="/home/$user/distro"
ROOTBUILDDIR="$ROOT/build"                # Derleme dizini
BUILDDIR="$ROOT/build/build-${name}-${version}" # Alt dizin
DESTDIR="$ROOT/rootfs"                     # Yükleme dizini
PACKAGEDIR=$(pwd)                          # Paket dizini
SOURCEDIR="$ROOT/build/${name}-${version}" # Kaynak dizini

initsetup(){
        mkdir -p  $ROOTBUILDDIR #derleme dizini yoksa oluşturuluyor
        rm -rf $ROOTBUILDDIR/* #içeriği temizleniyor
        cd $ROOTBUILDDIR #dizinine geçiyoruz
        wget ${source}
        #isimde boşluk varsa silme işlemi yapılıyor
        for f in *\ *; do mv "$f" "${f// /}"; done
        dowloadfile=$(ls|head -1)
        filetype=$(file -b --extension $dowloadfile|cut -d'/' -f1)
        if [ "${filetype}" == "???" ]; then unzip  ${dowloadfile}; \
        else tar -xvf ${dowloadfile};fi
        director=$(find ./* -maxdepth 0 -type d)
        directorname=$(basename ${director})
        if [ "${directorname}" != "${name}-${version}" ]; \
        then mv $directorname ${name}-${version};fi
        mkdir -p $BUILDDIR&&mkdir -p $DESTDIR&&cd $SOURCEDIR
}
setup() {
./configure --prefix=/usr --libdir=/usr/lib64 --bindir=/bin
--with-curses --enable-readline --without-bash-malloc
}
build() {
         make
}
package()       {
        make install DESTDIR=$DESTDIR
        cd $DESTDIR/bin
        ln -s bash sh
        ${DESTDIR}/sbin/ldconfig -r ${DESTDIR}  # sistem guncelleniyor
}
initsetup       # initsetup fonksiyonunu çalıştırır ve kaynak dosyayı indirir
setup           # setup fonksiyonu çalışır ve derleme öncesi kaynak dosyaların ayalanması sağlanır.
build           # build fonksiyonu çalışır ve kaynak dosyaları derlenir.
package         # package fonksiyonu çalışır, yükleme öncesi ayarlamalar yapılır ve yüklenir.
```

Paket Derleme Yöntemi konusunda anlatıldığı gibi derleme işlemini yapınız.

openssl

OpenSSL, şifreleme, sertifika yönetimi ve güvenli bağlantılar (TLS/SSL) oluşturmak için kullanılır. coreutils için gerekli olan paket.

Derleme

Debian ortamında bu paketin derlenmesi için; **sudo apt install perl** komutuyla paketin kurulması gerekmektedir.

```
#-------------------------------------------------------------------------------------------------------------------
#!/usr/bin/env bash
version="3.2.0"
name="openssl"
depends="glibc,zlib"
description="openssl"
source="https://www.openssl.org/source/${name}-${version}.tar.gz"

# Display adı ve kullanıcı
display=":$(ls /tmp/.X11-unix/* | sed 's#/tmp/.X11-unix/X##' | head -1)"
user=$(who | grep "(${display})" | awk '{print $1}')

ROOT="/home/$user/distro"
ROOTBUILDDIR="$ROOT/build"                    # Derleme dizini
BUILDDIR="$ROOT/build/build-${name}-${version}"  # Alt dizin
DESTDIR="$ROOT/rootfs"                         # Yükleme dizini
PACKAGEDIR=$(pwd)                             # Paket dizini
SOURCEDIR="$ROOT/build/${name}-${version}"   # Kaynak dizini

initsetup(){
        mkdir -p  $ROOTBUILDDIR #derleme dizini yoksa oluşturuluyor
        rm -rf $ROOTBUILDDIR/* #içeriği temizleniyor
        cd $ROOTBUILDDIR #dizinine geçiyoruz
        wget ${source}
        #isimde boşluk varsa silme işlemi yapılıyor
        for f in *\ *; do mv "$f" "${f// /}"; done
        dowloadfile=$(ls|head -1)
        filetype=$(file -b --extension $dowloadfile|cut -d'/' -f1)
        if [ "${filetype}" == "???" ]; then unzip  ${dowloadfile}; \
        else tar -xvf ${dowloadfile};fi
        director=$(find ./* -maxdepth 0 -type d)
        directorname=$(basename ${director})
        if [ "${directorname}" != "${name}-${version}" ]; \
        then mv $directorname ${name}-${version};fi
        mkdir -p $BUILDDIR&&mkdir -p $DESTDIR&&cd $SOURCEDIR
}

setup(){
        cp -prfv $PACKAGEDIR/files/ $SOURCEDIR
        wget -O "$SOURCEDIR/files/cacert.pem" https://curl.haxx.se/ca/cacert.pem
        patch -Np1 -i "$SOURCEDIR/files/ca-dir.patch"
        ./config --prefix=/usr --openssldir=/etc/ssl \
            --libdir=/usr/lib64 shared linux-x86_64
}

build(){
        make depend
        make
}

package(){
        mkdir -p "${DESTDIR}/etc/ssl/" "${DESTDIR}/sbin/"
        install "$SOURCEDIR/files/update-certdata" "${DESTDIR}/sbin/update-certdata"
        install "$SOURCEDIR/files/cacert.pem" "${DESTDIR}/etc/ssl/cert.pem"
        make DESTDIR="${DESTDIR}" install_sw install_ssldirs install_man_docs
        "${DESTDIR}/sbin/ldconfig" -r "${DESTDIR}"    # sistem güncelleniyor
}

initsetup       # initsetup fonksiyonunu çalıştırır ve kaynak dosyayı indirir
setup           # setup fonksiyonu çalışır ve derleme öncesi kaynak dosyaların ayalanması sağlanır.
build           # build fonksiyonu çalışır ve kaynak dosyaları derlenir.
package         # package fonksiyonu çalışır, yükleme öncesi ayarlamalar yapılır ve yüklenir.
```

Ek dosyaları indirmek için tıklayınız. **Paket Derleme Yöntemi** konusunda anlatıldığı gibi derleme işlemini yapınız.

acl

ACL (Access Control List), dosya ve sistemlerde kimlerin hangi işlemleri yapabileceğini belirleyen bir yapıdır. Erişim kontrolü daha düzenli ve güvenli hale gelir. ACL, özellikle coreutils gibi temel sistem araçlarının doğru çalışması için gerekli olan bileşenlerden biridir.

Derleme

Debian ortamında bu paketin derlenmesi için; **sudo apt install libattr1-dev** komutuyla paketin kurulması gerekmektedir.

```
#-----------------------------------------------------------------------------------------------------------------------
#!/usr/bin/env bash
name="acl"
version="2.3.1"
description="The acl package contains utilities to administer Access Control Lists"
source="https://download.savannah.nongnu.org/releases/acl/acl-$version.tar.xz"
depends="attr"
display=":$(ls /tmp/.X11-unix/* | sed 's#/tmp/.X11-unix/X##' | head -1)"  # Display adı
user=$(who | grep "(${display})" | awk '{print $1}')                      # Display kullanıcısı

ROOT="/home/$user/distro"
ROOTBUILDDIR="$ROOT/build"                # Derleme dizini
BUILDDIR="$ROOT/build/build-${name}-${version}" # Alt dizin
DESTDIR="$ROOT/rootfs"                     # Yükleme dizini
PACKAGEDIR=$(pwd)                          # Paket dizini
SOURCEDIR="$ROOT/build/${name}-${version}" # Kaynak dizini

initsetup(){
        mkdir -p  $ROOTBUILDDIR #derleme dizini yoksa oluşturuluyor
        rm -rf $ROOTBUILDDIR/* #içeriği temizleniyor
        cd $ROOTBUILDDIR #dizinine geçiyoruz
        wget ${source}
        #isimde boşluk varsa silme işlemi yapılıyor
        for f in *\ *; do mv "$f" "${f// /}"; done
        dowloadfile=$(ls|head -1)
        filetype=$(file -b --extension $dowloadfile|cut -d'/' -f1)
        if [ "${filetype}" == "???" ]; then unzip  ${dowloadfile}; \
        else tar -xvf ${dowloadfile};fi
        director=$(find ./* -maxdepth 0 -type d)
        directorname=$(basename ${director})
        if [ "${directorname}" != "${name}-${version}" ]; \
        then mv $directorname ${name}-${version};fi
        mkdir -p $BUILDDIR&&mkdir -p $DESTDIR&&cd $SOURCEDIR
}
setup(){
    ./configure --prefix=/usr --libdir=/usr/lib64
}
build(){
     make
}
package(){
    make install DESTDIR=$DESTDIR
    ${DESTDIR}/sbin/ldconfig -r ${DESTDIR}           # sistem guncelleniyor
}

initsetup       # initsetup fonksiyonunu çalıştırır ve kaynak dosyayı indirir
setup           # setup fonksiyonu çalışır ve derleme öncesi kaynak dosyaların ayalanması sağlanır.
build           # build fonksiyonu çalışır ve kaynak dosyaları derlenir.
package         # package fonksiyonu çalışır, yükleme öncesi ayarlamalar yapılır ve yüklenir.
```

Paket Derleme Yöntemi konusunda anlatıldığı gibi derleme işlemini yapınız.

attr

coreutils için gerekli olan paket. attr, dosya özniteliklerini ayarlamak veya görüntülemek için kullanılan bir komuttur. Bu komut, dosya veya dizinlerin özelliklerini (izinler, sahiplik, erişim zamanları vb.) yönetmek için kullanılır.

Derleme

Debian ortamında bu paketin derlenmesi için; **sudo apt install libattr1-dev** komutuyla paketin kurulması gerekmektedir.

```
#-----------------------------------------------------------------------------------------------------------------------
#!/usr/bin/env bash
name="attr"
version="2.5.1"
description="The attr package contains utilities to administer the extended attributes on filesystem objects."
source="https://mirror.rabisu.com/savannah-nongnu/attr/attr-${version}.tar.gz"
depends=""
display=":$(ls /tmp/.X11-unix/* | sed 's#/tmp/.X11-unix/X##' | head -1)"  # Display adı
user=$(who | grep "(${display})" | awk '{print $1}')                      # Display kullanıcısı

ROOT="/home/$user/distro"
ROOTBUILDDIR="$ROOT/build"                # Derleme dizini
BUILDDIR="$ROOT/build/build-${name}-${version}" # Alt dizin
DESTDIR="$ROOT/rootfs"                     # Yükleme dizini
PACKAGEDIR=$(pwd)                          # Paket dizini
SOURCEDIR="$ROOT/build/${name}-${version}" # Kaynak dizini

initsetup(){
        mkdir -p  $ROOTBUILDDIR #derleme dizini yoksa oluşturuluyor
        rm -rf $ROOTBUILDDIR/* #içeriği temizleniyor
        cd $ROOTBUILDDIR #dizinine geçiyoruz
        wget ${source}
        #isimde boşluk varsa silme işlemi yapılıyor
        for f in *\ *; do mv "$f" "${f// /}"; done
        dowloadfile=$(ls|head -1)
        filetype=$(file -b --extension $dowloadfile|cut -d'/' -f1)
        if [ "${filetype}" == "???" ]; then unzip  ${dowloadfile}; \
        else tar -xvf ${dowloadfile};fi
        director=$(find ./* -maxdepth 0 -type d)
        directorname=$(basename ${director})
        if [ "${directorname}" != "${name}-${version}" ]; \
        then mv $directorname ${name}-${version};fi
        mkdir -p $BUILDDIR&&mkdir -p $DESTDIR&&cd $SOURCEDIR
}
setup(){
    ./configure --prefix=/usr --sysconfdir=/etc --libdir=/usr/lib64
}

build(){
    make
}

package(){
    make install DESTDIR=$DESTDIR
    ${DESTDIR}/sbin/ldconfig -r ${DESTDIR}           # sistem guncelleniyor
}

initsetup       # initsetup fonksiyonunu çalıştırır ve kaynak dosyayı indirir
setup           # setup fonksiyonu çalışır ve derleme öncesi kaynak dosyaların ayalanması sağlanır.
build           # build fonksiyonu çalışır ve kaynak dosyaları derlenir.
package         # package fonksiyonu çalışır, yükleme öncesi ayarlamalar yapılır ve yüklenir.
```

Paket Derleme Yöntemi konusunda anlatıldığı gibi derleme işlemini yapınız.

libcap

libcap paketi, Linux işletim sistemlerinde kullanıcı ve grup yetkilendirmelerini yönetmek için kullanılan bir kütüphanedir. Bu kütüphane, sistem kaynaklarına erişim kontrolü sağlamak amacıyla çeşitli yetki seviyeleri tanımlamaktadır. coreutils için gerekli olan paket.

Derleme

```
#-------------------------------------------------------------------------------------------------------------------------
#!/usr/bin/env bash
version="2.69"
name="libcap"
depends="glibc,acl,openssl"
description="shell ve network copy"
source="https://mirrors.edge.kernel.org/pub/linux/libs/security/linux-privs/libcap2/${name}-${version}.tar.xz"
display=":$(ls /tmp/.X11-unix/* | sed 's#/tmp/.X11-unix/X##' | head -1)"  # Display adı
user=$(who | grep "(${display})" | awk '{print $1}')                      # Display kullanıcısı

ROOT="/home/$user/distro"
ROOTBUILDDIR="$ROOT/build"                # Derleme dizini
BUILDDIR="$ROOT/build/build-${name}-${version}" # Alt dizin
DESTDIR="$ROOT/rootfs"                     # Yükleme dizini
PACKAGEDIR=$(pwd)                          # Paket dizini
SOURCEDIR="$ROOT/build/${name}-${version}" # Kaynak dizini

initsetup(){
        mkdir -p  $ROOTBUILDDIR #derleme dizini yoksa oluşturuluyor
        rm -rf $ROOTBUILDDIR/* #içeriği temizleniyor
        cd $ROOTBUILDDIR #dizinine geçiyoruz
        wget ${source}
        #isimde boşluk varsa silme işlemi yapılıyor
        for f in *\ *; do mv "$f" "${f// /}"; done
        dowloadfile=$(ls|head -1)
        filetype=$(file -b --extension $dowloadfile|cut -d'/' -f1)
        if [ "${filetype}" == "???" ]; then unzip  ${dowloadfile}; \
        else tar -xvf ${dowloadfile};fi
        director=$(find ./* -maxdepth 0 -type d)
        directorname=$(basename ${director})
        if [ "${directorname}" != "${name}-${version}" ]; \
        then mv $directorname ${name}-${version};fi
        mkdir -p $BUILDDIR&&mkdir -p $DESTDIR&&cd $SOURCEDIR
}
_common_make_options=(
  CGO_CPPFLAGS="$CPPFLAGS"
  CGO_CFLAGS="$CFLAGS"
  CGO_CXXFLAGS="$CXXFLAGS"
  CGO_LDFLAGS="$LDFLAGS"
  CGO_REQUIRED="1"
  GOFLAGS="-buildmode=pie -mod=readonly -modcacherw"
  GO_BUILD_FLAGS="-ldflags '-compressdwarf=false -linkmode=external'"
)
setup(){ echo ""
}
build(){
make "${_common_make_options[@]}" SUDO="" prefix=/usr KERNEL_HEADERS=include lib=lib64 \
sbindir=bin RAISE_SETFCAP=no  DYNAMIC=yes
}
package(){
        make "${_common_make_options[@]}" DESTDIR="$DESTDIR" RAISE_SETFCAP=no prefix=/usr \
        lib=lib64 sbindir=bin install
        ${DESTDIR}/sbin/ldconfig -r ${DESTDIR}           # sistem guncelleniyor
}

initsetup       # initsetup fonksiyonunu çalıştırır ve kaynak dosyayı indirir
setup           # setup fonksiyonu çalışır ve derleme öncesi kaynak dosyaların ayalanması sağlanır.
build           # build fonksiyonu çalışır ve kaynak dosyaları derlenir.
package         # package fonksiyonu çalışır, yükleme öncesi ayarlamalar yapılır ve yüklenir.
```

Paket Derleme Yöntemi konusunda anlatıldığı gibi derleme işlemini yapınız.

libpcre2

libpcre2 paketi, Perl Compatible Regular Expressions (PCRE) kütüphanesinin ikinci versiyonunu temsil eder ve düzenli ifadelerle çalışmak için kullanılan bir araçtır. Bu kütüphane, özellikle metin işleme ve arama işlemlerinde yaygın olarak kullanılmaktadır.

Derleme

```
#-----------------------------------------------------------------------------------------------------------------------
#!/usr/bin/env bash
version="10.40"
name="libpcre2"
description="Perl-compatible regular expression library"
source="https://github.com/PCRE2Project/pcre2/releases/download/pcre2-${version}/\
pcre2-${version}.tar.gz"
depends="readline"
display=":$(ls /tmp/.X11-unix/* | sed 's#/tmp/.X11-unix/X##' | head -1)"  # Display adı
user=$(who | grep "(${display})" | awk '{print $1}')                      # Display kullanıcısı

ROOT="/home/$user/distro"
ROOTBUILDDIR="$ROOT/build"                # Derleme dizini
BUILDDIR="$ROOT/build/build-${name}-${version}" # Alt dizin
DESTDIR="$ROOT/rootfs"                     # Yükleme dizini
PACKAGEDIR=$(pwd)                          # Paket dizini
SOURCEDIR="$ROOT/build/${name}-${version}" # Kaynak dizini

initsetup(){
        mkdir -p  $ROOTBUILDDIR #derleme dizini yoksa oluşturuluyor
        rm -rf $ROOTBUILDDIR/* #içeriği temizleniyor
        cd $ROOTBUILDDIR #dizinine geçiyoruz
        wget ${source}
        #isimde boşluk varsa silme işlemi yapılıyor
        for f in *\ *; do mv "$f" "${f// /}"; done
        dowloadfile=$(ls|head -1)
        filetype=$(file -b --extension $dowloadfile|cut -d'/' -f1)
        if [ "${filetype}" == "???" ]; then unzip  ${dowloadfile}; \
        else tar -xvf ${dowloadfile};fi
        director=$(find ./* -maxdepth 0 -type d)
        directorname=$(basename ${director})
        if [ "${directorname}" != "${name}-${version}" ]; \
        then mv $directorname ${name}-${version};fi
        mkdir -p $BUILDDIR&&mkdir -p $DESTDIR&&cd $SOURCEDIR
}
setup() {
    ./configure --prefix=/usr --enable-shared --enable-static --enable-pcre2-16 \
    --enable-pcre2-32 --enable-jit --enable-pcre2test-libreadline
}
build() {
        make
}
package(){
        make install DESTDIR=$DESTDIR
        ${DESTDIR}/sbin/ldconfig -r ${DESTDIR}          # sistem guncelleniyor
}

initsetup       # initsetup fonksiyonunu çalıştırır ve kaynak dosyayı indirir
setup           # setup fonksiyonu çalışır ve derleme öncesi kaynak dosyaların ayalanması sağlanır.
build           # build fonksiyonu çalışır ve kaynak dosyaları derlenir.
package         # package fonksiyonu çalışır, yükleme öncesi ayarlamalar yapılır ve yüklenir.
```

Paket Derleme Yöntemi konusunda anlatıldığı gibi derleme işlemini yapınız.

gmp

GMP (GNU Multiple Precision Arithmetic Library) paketi, yüksek hassasiyetli aritmetik işlemler gerçekleştirmek için kullanılan bir kütüphanedir. Özellikle büyük sayılarla çalışırken, standart veri türlerinin yetersiz kaldığı durumlarda devreye girer.

Derleme

```
#-----------------------------------------------------------------------------------------------------------------------
#!/usr/bin/env bash
version="6.3.0"
name="gmp"
depends="glibc"
description="Library for arbitrary-precision arithmetic on different type of numbers"
source="https://gmplib.org/download/gmp/gmp-${version}.tar.xz"
display=":$(ls /tmp/.X11-unix/* | sed 's#/tmp/.X11-unix/X##' | head -1)"  # Display adı
user=$(who | grep "(${display})" | awk '{print $1}')                      # Display kullanıcısı

ROOT="/home/$user/distro"
ROOTBUILDDIR="$ROOT/build"                # Derleme dizini
BUILDDIR="$ROOT/build/build-${name}-${version}" # Alt dizin
DESTDIR="$ROOT/rootfs"                     # Yükleme dizini
PACKAGEDIR=$(pwd)                          # Paket dizini
SOURCEDIR="$ROOT/build/${name}-${version}" # Kaynak dizini

initsetup(){
            mkdir -p  $ROOTBUILDDIR #derleme dizini yoksa oluşturuluyor
            rm -rf $ROOTBUILDDIR/* #içeriği temizleniyor
            cd $ROOTBUILDDIR #dizinine geçiyoruz
    wget ${source}
    for f in *\ *; do mv "$f" "${f// /}"; done #isimde boşluk varsa silme işlemi yapılıyor
            dowloadfile=$(ls|head -1)
            filetype=$(file -b --extension $dowloadfile|cut -d'/' -f1)
            if [ "${filetype}" == "???" ]; then unzip  ${dowloadfile}; else tar -xvf ${dowloadfile};fi
            director=$(find ./* -maxdepth 0 -type d)
            directorname=$(basename ${director})
            if [ "${directorname}" != "${name}-${version}" ]; then mv $directorname ${name}-${version};fi
            mkdir -p $BUILDDIR&&mkdir -p $DESTDIR&&cd $SOURCEDIR
}

setup()
{
        ./configure --prefix=/usr --libdir=/usr/lib64 \
                $(use_opt cxx --enable-cxx --disable-cxx) \
        --enable-fat
}
build()
{
        make
}
package()
{
        make install DESTDIR=${DESTDIR}
        ${DESTDIR}/sbin/ldconfig -r ${DESTDIR}           # sistem guncelleniyor
}
initsetup       # initsetup fonksiyonunu çalıştırır ve kaynak dosyayı indirir
setup           # setup fonksiyonu çalışır ve derleme öncesi kaynak dosyaların ayalanması sağlanır.
build           # build fonksiyonu çalışır ve kaynak dosyaları derlenir.
package         # package fonksiyonu çalışır, yükleme öncesi ayarlamalar yapılır ve yüklenir.
```

Paket Derleme Yöntemi konusunda anlatıldığı gibi derleme işlemini yapınız.

coreutils

coreutils, Linux işletim sistemlerinde temel dosya yönetimi ve sistem komutlarını içeren bir paket olup, kullanıcıların dosyalarla etkileşimde bulunmalarını sağlayan temel araçları sunar. Bu paket, dosya kopyalama, taşıma, silme, listeleme gibi işlemleri gerçekleştiren komutları içerir. Örneğin, cp, mv, rm, ls gibi komutlar coreutils paketinin bir parçasıdır.

Derleme

```
#-----------------------------------------------------------------------------------------------------------------------
#!/usr/bin/env bash
name="coreutils"
version="9.7"
description="The basic file, shell and text manipulation utilities of the GNU operating system"
source="https://ftp.gnu.org/gnu/coreutils/coreutils-$version.tar.xz"
depends="acl,attr,gmp,libcap,openssl"
display=":$(ls /tmp/.X11-unix/* | sed 's#/tmp/.X11-unix/X##' | head -1)"  # Display adı
user=$(who | grep "(${display})" | awk '{print $1}')                      # Display kullanıcısı

ROOT="/home/$user/distro"
ROOTBUILDDIR="$ROOT/build"                # Derleme dizini
BUILDDIR="$ROOT/build/build-${name}-${version}" # Alt dizin
DESTDIR="$ROOT/rootfs"                     # Yükleme dizini
PACKAGEDIR=$(pwd)                          # Paket dizini
SOURCEDIR="$ROOT/build/${name}-${version}" # Kaynak dizini

initsetup(){
            mkdir -p  $ROOTBUILDDIR #derleme dizini yoksa oluşturuluyor
            rm -rf $ROOTBUILDDIR/* #içeriği temizleniyor
            cd $ROOTBUILDDIR #dizinine geçiyoruz
    wget ${source}
    for f in *\ *; do mv "$f" "${f// /}"; done #isimde boşluk varsa silme işlemi yapılıyor
            dowloadfile=$(ls|head -1)
            filetype=$(file -b --extension $dowloadfile|cut -d'/' -f1)
            if [ "${filetype}" == "???" ]; then unzip  ${dowloadfile}; else tar -xvf ${dowloadfile};fi
            director=$(find ./* -maxdepth 0 -type d)
            directorname=$(basename ${director})
            if [ "${directorname}" != "${name}-${version}" ]; then mv $directorname ${name}-${version};fi
            mkdir -p $BUILDDIR&&mkdir -p $DESTDIR&&cd $SOURCEDIR
}

export CFLAGS="-static-libgcc -static-libstdc++ -fPIC"
setup(){
export FORCE_UNSAFE_CONFIGURE=1
./configure --prefix=/usr \
--libdir=/usr/lib64 \
--libexecdir=/usr/libexec \
--enable-largefile \
--enable-single-binary=symlinks \
--enable-no-install-program=groups,hostname,kill,uptime \
--without-selinux --without-openssl
}

build(){
    make
}
package(){
    make install DESTDIR=$DESTDIR
    ${DESTDIR}/sbin/ldconfig -r ${DESTDIR}           # sistem guncelleniyor
}
initsetup       # initsetup fonksiyonunu çalıştırır ve kaynak dosyayı indirir
setup           # setup fonksiyonu çalışır ve derleme öncesi kaynak dosyaların ayalanması sağlanır.
build           # build fonksiyonu çalışır ve kaynak dosyaları derlenir.
package         # package fonksiyonu çalışır, yükleme öncesi ayarlamalar yapılır ve yüklenir.
```

Paket Derleme Yöntemi konusunda anlatıldığı gibi derleme işlemini yapınız.

sqlite

SQLite, bir veritabanı sistemidir. Sunucu gerektirmez ve verileri tek bir dosya içinde saklar.

Derleme öncesi **sudo apt install libsqlite3-dev** komutuyla paketi kurmalıyız.

Derleme

```
#-----------------------------------------------------------------------------------------------------------------------
#!/usr/bin/env bash
name="sqlite"
version="3.45.2"
description="A C library that implements an SQL database engine"
srcver="3450200"
source="https://www.sqlite.org/2024/sqlite-autoconf-$srcver.tar.gz"
depends="zlib,readline"
display=":$(ls /tmp/.X11-unix/* | sed 's#/tmp/.X11-unix/X##' | head -1)"  # Display adı
user=$(who | grep "(${display})" | awk '{print $1}')                      # Display kullanıcısı

ROOT="/home/$user/distro"
ROOTBUILDDIR="$ROOT/build"                # Derleme dizini
BUILDDIR="$ROOT/build/build-${name}-${version}" # Alt dizin
DESTDIR="$ROOT/rootfs"                     # Yükleme dizini
PACKAGEDIR=$(pwd)                          # Paket dizini
SOURCEDIR="$ROOT/build/${name}-${version}" # Kaynak dizini


initsetup(){
            mkdir -p  $ROOTBUILDDIR #derleme dizini yoksa oluşturuluyor
            rm -rf $ROOTBUILDDIR/* #içeriği temizleniyor
            cd $ROOTBUILDDIR #dizinine geçiyoruz
    wget ${source}
    for f in *\ *; do mv "$f" "${f// /}"; done #isimde boşluk varsa silme işlemi yapılıyor
            dowloadfile=$(ls|head -1)
            filetype=$(file -b --extension $dowloadfile|cut -d'/' -f1)
            if [ "${filetype}" == "???" ]; then unzip  ${dowloadfile}; else tar -xvf ${dowloadfile};fi
            director=$(find ./* -maxdepth 0 -type d)
            directorname=$(basename ${director})
            if [ "${directorname}" != "${name}-${version}" ]; then mv $directorname ${name}-${version};fi
            mkdir -p $BUILDDIR&&mkdir -p $DESTDIR&&cd $SOURCEDIR
}


setup(){
    ./configure --prefix=/usr --libdir=/usr/lib64 --enable-fts3 --enable-fts4 --enable-fts5 --enable-rtree \
        CPPFLAGS="-DSQLITE_ENABLE_FTS3=1            \
                  -DSQLITE_ENABLE_FTS4=1            \
                  -DSQLITE_ENABLE_COLUMN_METADATA=1 \
                  -DSQLITE_ENABLE_UNLOCK_NOTIFY=1   \
                  -DSQLITE_ENABLE_DBSTAT_VTAB=1     \
                  -DSQLITE_SECURE_DELETE=1          \
                  -DSQLITE_ENABLE_FTS3_TOKENIZER=1"
}

build(){
    make
}

package(){
    make install DESTDIR=$DESTDIR
    ${DESTDIR}/sbin/ldconfig -r ${DESTDIR}           # sistem guncelleniyor
}
initsetup       # initsetup fonksiyonunu çalıştırır ve kaynak dosyayı indirir
setup           # setup fonksiyonu çalışır ve derleme öncesi kaynak dosyaların ayalanması sağlanır.
build           # build fonksiyonu çalışır ve kaynak dosyaları derlenir.
package         # package fonksiyonu çalışır, yükleme öncesi ayarlamalar yapılır ve yüklenir.
```

Paket Derleme Yöntemi konusunda anlatıldığı gibi derleme işlemini yapınız.

util-linux

util-linux, Linux sistemlerinde kullanılan temel araçları içeren pakettir. Dosya sistemlerini bağlamak/ayırmak, parted araçları, kullanıcı işlemleri ve kimlik doğrulama için kullanılır.

Debian ortamında bu paketin derlenmesi için; **sudo apt install libudev-dev** komutuyla paketin kurulması gerekmektedir.

Derleme

```
#-----------------------------------------------------------------------------------------------------------------------
#!/usr/bin/env bash
name="util-linux"
version="2.40.1"
description="Various useful Linux utilities"
source="https://mirrors.edge.kernel.org/pub/linux/utils/util-linux/v${version%.*}/util-linux-${version}.tar.gz"
depends=""
buildepend="libcap-ng,python,eudev,sqlite,eudev,cryptsetup,libxcrypt"
display=":$(ls /tmp/.X11-unix/* | sed 's#/tmp/.X11-unix/X##' | head -1)"  # Display adı
user=$(who | grep "(${display})" | awk '{print $1}')                      # Display kullanıcısı

ROOT="/home/$user/distro"
ROOTBUILDDIR="$ROOT/build"                # Derleme dizini
BUILDDIR="$ROOT/build/build-${name}-${version}" # Alt dizin
DESTDIR="$ROOT/rootfs"                     # Yükleme dizini
PACKAGEDIR=$(pwd)                          # Paket dizini
SOURCEDIR="$ROOT/build/${name}-${version}" # Kaynak dizini

initsetup(){
        mkdir -p  $ROOTBUILDDIR #derleme dizini yoksa oluşturuluyor
        rm -rf $ROOTBUILDDIR/* #içeriği temizleniyor
        cd $ROOTBUILDDIR #dizinine geçiyoruz
        wget ${source}
        for f in *\ *; do mv "$f" "${f// /}"; done #isimde boşluk varsa silme işlemi yapılıyor
        dowloadfile=$(ls|head -1)
        filetype=$(file -b --extension $dowloadfile|cut -d'/' -f1)
        if [ "${filetype}" == "???" ]; then unzip  ${dowloadfile}; else tar -xvf ${dowloadfile};fi
        director=$(find ./* -maxdepth 0 -type d)
        directorname=$(basename ${director})
        if [ "${directorname}" != "${name}-${version}" ]; then mv $directorname ${name}-${version};fi
        mkdir -p $BUILDDIR&&mkdir -p $DESTDIR&&cd $SOURCEDIR
}
setup(){
        cp -prvf $PACKAGEDIR/files/ $SOURCEDIR/
        patch -Np1 -i $SOURCEDIR/files/0001-util-linux-tmpfiles.patch
        ./configure --prefix=/usr --libdir=/usr/lib64 --bindir=/usr/bin --enable-shared --enable-static \
        --disable-su --disable-runuser --disable-chfn-chsh --disable-login --disable-sulogin  \
        --disable-makeinstall-chown --disable-makeinstall-setuid --disable-pylibmount \
        --disable-raw --without-systemd --without-libuser --without-utempter --without-econf \
        --with-python --with-udev
}
build(){
        make
}
package(){
        make install DESTDIR=$DESTDIR
        ${DESTDIR}/sbin/ldconfig -r ${DESTDIR}          # sistem guncelleniyor
}
initsetup       # initsetup fonksiyonunu çalıştırır ve kaynak dosyayı indirir
setup           # setup fonksiyonu çalışır ve derleme öncesi kaynak dosyaların ayalanması sağlanır.
build           # build fonksiyonu çalışır ve kaynak dosyaları derlenir.
package         # package fonksiyonu çalışır, yükleme öncesi ayarlamalar yapılır ve yüklenir.
```

Ek dosyaları indirmek için tıklayınız. **Paket Derleme Yöntemi** konusunda anlatıldığı gibi derleme işlemini yapınız.

grep

Grep, Linux işletim sistemlerinde metin arama işlemleri için kullanılan güçlü bir komut satırı aracıdır. Kullanıcıların belirli bir desen veya kelimeyi dosyalar içinde hızlı bir şekilde bulmalarını sağlar.

Derleme

```
#-----------------------------------------------------------------------------------------------------------------------
#!/usr/bin/env bash
name="grep"
version="3.11"
description="GNU regular expression matcher"
source="https://ftp.gnu.org/gnu/grep/grep-$version.tar.xz"
depends="libpcre2"
display=":$(ls /tmp/.X11-unix/* | sed 's#/tmp/.X11-unix/X##' | head -1)"  # Display adı
user=$(who | grep "(${display})" | awk '{print $1}')                      # Display kullanıcısı

ROOT="/home/$user/distro"
ROOTBUILDDIR="$ROOT/build"                # Derleme dizini
BUILDDIR="$ROOT/build/build-${name}-${version}" # Alt dizin
DESTDIR="$ROOT/rootfs"                     # Yükleme dizini
PACKAGEDIR=$(pwd)                          # Paket dizini
SOURCEDIR="$ROOT/build/${name}-${version}" # Kaynak dizini

initsetup(){
            mkdir -p  $ROOTBUILDDIR #derleme dizini yoksa oluşturuluyor
            rm -rf $ROOTBUILDDIR/* #içeriği temizleniyor
            cd $ROOTBUILDDIR #dizinine geçiyoruz
    wget ${source}
    for f in *\ *; do mv "$f" "${f// /}"; done #isimde boşluk varsa silme işlemi yapılıyor
            dowloadfile=$(ls|head -1)
            filetype=$(file -b --extension $dowloadfile|cut -d'/' -f1)
            if [ "${filetype}" == "???" ]; then unzip  ${dowloadfile}; else tar -xvf ${dowloadfile};fi
            director=$(find ./* -maxdepth 0 -type d)
            directorname=$(basename ${director})
            if [ "${directorname}" != "${name}-${version}" ]; then mv $directorname ${name}-${version};fi
            mkdir -p $BUILDDIR&&mkdir -p $DESTDIR&&cd $SOURCEDIR
}

setup(){
        ./configure --prefix=/usr --libdir=/usr/lib64/
}

build(){
    make
}

package(){
    make install DESTDIR=$DESTDIR
    ${DESTDIR}/sbin/ldconfig -r ${DESTDIR}           # sistem guncelleniyor
}
initsetup       # initsetup fonksiyonunu çalıştırır ve kaynak dosyayı indirir
setup           # setup fonksiyonu çalışır ve derleme öncesi kaynak dosyaların ayalanması sağlanır.
build           # build fonksiyonu çalışır ve kaynak dosyaları derlenir.
package         # package fonksiyonu çalışır, yükleme öncesi ayarlamalar yapılır ve yüklenir.
```

Paket Derleme Yöntemi konusunda anlatıldığı gibi derleme işlemini yapınız.

sed

sed, "stream editor" anlamına gelen bir Linux komut satırı aracıdır. sed, dosyalar üzerinde arama, değiştirme, silme ve ekleme işlemleri yaparak metin verilerini işlemek için kullanılır.

Derleme

```
#-----------------------------------------------------------------------------------------------------------------------
#!/usr/bin/env bash
name="sed"
version="4.9"
description="Super-useful stream editor"
source="https://ftp.gnu.org/gnu/sed/sed-$version.tar.xz"
depends="acl"
display=":$(ls /tmp/.X11-unix/* | sed 's#/tmp/.X11-unix/X##' | head -1)"  # Display adı
user=$(who | grep "(${display})" | awk '{print $1}')                      # Display kullanıcısı

ROOT="/home/$user/distro"
ROOTBUILDDIR="$ROOT/build"                # Derleme dizini
BUILDDIR="$ROOT/build/build-${name}-${version}" # Alt dizin
DESTDIR="$ROOT/rootfs"                     # Yükleme dizini
PACKAGEDIR=$(pwd)                          # Paket dizini
SOURCEDIR="$ROOT/build/${name}-${version}" # Kaynak dizini


initsetup(){
            mkdir -p  $ROOTBUILDDIR #derleme dizini yoksa oluşturuluyor
            rm -rf $ROOTBUILDDIR/* #içeriği temizleniyor
            cd $ROOTBUILDDIR #dizinine geçiyoruz
    wget ${source}
    for f in *\ *; do mv "$f" "${f// /}"; done #isimde boşluk varsa silme işlemi yapılıyor
            dowloadfile=$(ls|head -1)
            filetype=$(file -b --extension $dowloadfile|cut -d'/' -f1)
            if [ "${filetype}" == "???" ]; then unzip  ${dowloadfile}; else tar -xvf ${dowloadfile};fi
            director=$(find ./* -maxdepth 0 -type d)
            directorname=$(basename ${director})
            if [ "${directorname}" != "${name}-${version}" ]; then mv $directorname ${name}-${version};fi
            mkdir -p $BUILDDIR&&mkdir -p $DESTDIR&&cd $SOURCEDIR
}

setup(){
   ./configure --prefix=/usr \
        --libdir=/usr/lib64/
}

build(){
    make
}

package(){
    make install DESTDIR=$DESTDIR
    ${DESTDIR}/sbin/ldconfig -r ${DESTDIR}           # sistem guncelleniyor
}
initsetup       # initsetup fonksiyonunu çalıştırır ve kaynak dosyayı indirir
setup           # setup fonksiyonu çalışır ve derleme öncesi kaynak dosyaların ayalanması sağlanır.
build           # build fonksiyonu çalışır ve kaynak dosyaları derlenir.
package         # package fonksiyonu çalışır, yükleme öncesi ayarlamalar yapılır ve yüklenir.
```

Paket Derleme Yöntemi konusunda anlatıldığı gibi derleme işlemini yapınız.

mpfr

MPFR (Multiple Precision Floating-Point Reliable) paketi, yüksek hassasiyetli kayan nokta hesaplamaları yapmak için kullanılan bir kütüphanedir.

Debian ortamında bu paketin derlenmesi için; **sudo apt install libgmp-dev** komutuyla paketin kurulması gerekmektedir.

Derleme

```
#-------------------------------------------------------------------------------------------------------------------
#!/usr/bin/env bash
version="4.2.0"
name="mpfr"
depends="glibc,gmp"
description="multiple precision floating-point computation"
source="https://ftp.gnu.org/gnu/mpfr/mpfr-${version}.tar.xz"
display=":$(ls /tmp/.X11-unix/* | sed 's#/tmp/.X11-unix/X##' | head -1)"  # Display adı
user=$(who | grep "(${display})" | awk '{print $1}')                      # Display kullanıcısı

ROOT="/home/$user/distro"
ROOTBUILDDIR="$ROOT/build"                # Derleme dizini
BUILDDIR="$ROOT/build/build-${name}-${version}" # Alt dizin
DESTDIR="$ROOT/rootfs"                     # Yükleme dizini
PACKAGEDIR=$(pwd)                          # Paket dizini
SOURCEDIR="$ROOT/build/${name}-${version}" # Kaynak dizini


initsetup(){
            mkdir -p  $ROOTBUILDDIR #derleme dizini yoksa oluşturuluyor
            rm -rf $ROOTBUILDDIR/* #içeriği temizleniyor
            cd $ROOTBUILDDIR #dizinine geçiyoruz
    wget ${source}
    for f in *\ *; do mv "$f" "${f// /}"; done #isimde boşluk varsa silme işlemi yapılıyor
            dowloadfile=$(ls|head -1)
            filetype=$(file -b --extension $dowloadfile|cut -d'/' -f1)
            if [ "${filetype}" == "???" ]; then unzip  ${dowloadfile}; else tar -xvf ${dowloadfile};fi
            director=$(find ./* -maxdepth 0 -type d)
            directorname=$(basename ${director})
            if [ "${directorname}" != "${name}-${version}" ]; then mv $directorname ${name}-${version};fi
            mkdir -p $BUILDDIR&&mkdir -p $DESTDIR&&cd $SOURCEDIR
}

setup()
{
        ./configure --prefix=/usr \
        --libdir=/usr/lib64 \
        --enable-shared \
        --enable-static \
        --disable-maintainer-mode \
        --enable-thread-safe
}
build()
{
        make
}
package()
{
        make install DESTDIR=${DESTDIR}
        ${DESTDIR}/sbin/ldconfig -r ${DESTDIR}          # sistem guncelleniyor
}
initsetup       # initsetup fonksiyonunu çalıştırır ve kaynak dosyayı indirir
setup           # setup fonksiyonu çalışır ve derleme öncesi kaynak dosyaların ayalanması sağlanır.
build           # build fonksiyonu çalışır ve kaynak dosyaları derlenir.
package         # package fonksiyonu çalışır, yükleme öncesi ayarlamalar yapılır ve yüklenir.
```

Paket Derleme Yöntemi konusunda anlatıldığı gibi derleme işlemini yapınız.

gawk

awk dilinin GNU versiyonu olan gawk, özellikle metin dosyalarındaki verileri analiz etmek, düzenlemek ve raporlamak için kullanılır.

Derleme

```
#-----------------------------------------------------------------------------------------------------------------------
#!/usr/bin/env bash
name="gawk"
version="5.3.0"
description="GNU awk pattern-matching language"
source="https://ftp.gnu.org/gnu/gawk/gawk-$version.tar.xz"
depends="mpfr,readline"
display=":$(ls /tmp/.X11-unix/* | sed 's#/tmp/.X11-unix/X##' | head -1)"  # Display adı
user=$(who | grep "(${display})" | awk '{print $1}')                      # Display kullanıcısı

ROOT="/home/$user/distro"
ROOTBUILDDIR="$ROOT/build"                # Derleme dizini
BUILDDIR="$ROOT/build/build-${name}-${version}" # Alt dizin
DESTDIR="$ROOT/rootfs"                     # Yükleme dizini
PACKAGEDIR=$(pwd)                          # Paket dizini
SOURCEDIR="$ROOT/build/${name}-${version}" # Kaynak dizini


initsetup(){
            mkdir -p  $ROOTBUILDDIR #derleme dizini yoksa oluşturuluyor
            rm -rf $ROOTBUILDDIR/* #içeriği temizleniyor
            cd $ROOTBUILDDIR #dizinine geçiyoruz
    wget ${source}
    for f in *\ *; do mv "$f" "${f// /}"; done #isimde boşluk varsa silme işlemi yapılıyor
            dowloadfile=$(ls|head -1)
            filetype=$(file -b --extension $dowloadfile|cut -d'/' -f1)
            if [ "${filetype}" == "???" ]; then unzip  ${dowloadfile}; else tar -xvf ${dowloadfile};fi
            director=$(find ./* -maxdepth 0 -type d)
            directorname=$(basename ${director})
            if [ "${directorname}" != "${name}-${version}" ]; then mv $directorname ${name}-${version};fi
            mkdir -p $BUILDDIR&&mkdir -p $DESTDIR&&cd $SOURCEDIR
}

setup(){
    ./configure --prefix=/usr \
        --libdir=/usr/lib64/ \
        --sysconfdir=/etc \
        --without-libsigsegv
}

build(){
    make
}

package(){
    make install DESTDIR=$DESTDIR
    ${DESTDIR}/sbin/ldconfig -r ${DESTDIR}          # sistem guncelleniyor
}
initsetup       # initsetup fonksiyonunu çalıştırır ve kaynak dosyayı indirir
setup           # setup fonksiyonu çalışır ve derleme öncesi kaynak dosyaların ayalanması sağlanır.
build           # build fonksiyonu çalışır ve kaynak dosyaları derlenir.
package         # package fonksiyonu çalışır, yükleme öncesi ayarlamalar yapılır ve yüklenir.
```

Paket Derleme Yöntemi konusunda anlatıldığı gibi derleme işlemini yapınız.

findutils

finutils paketi, dosya ve dizin yönetimi ile ilgili çeşitli işlevler sunan bir araç setidir.

Derleme

```
#-----------------------------------------------------------------------------------------------------------------------
#!/usr/bin/env bash
name="findutils"
version="4.9.0"
description="GNU utilities for finding files"
source="https://ftp.gnu.org/gnu/findutils/findutils-$version.tar.xz"
depends=""
display=":$(ls /tmp/.X11-unix/* | sed 's#/tmp/.X11-unix/X##' | head -1)"  # Display adı
user=$(who | grep "(${display})" | awk '{print $1}')                      # Display kullanıcısı

ROOT="/home/$user/distro"
ROOTBUILDDIR="$ROOT/build"                # Derleme dizini
BUILDDIR="$ROOT/build/build-${name}-${version}" # Alt dizin
DESTDIR="$ROOT/rootfs"                     # Yükleme dizini
PACKAGEDIR=$(pwd)                          # Paket dizini
SOURCEDIR="$ROOT/build/${name}-${version}" # Kaynak dizini


initsetup(){
            mkdir -p  $ROOTBUILDDIR #derleme dizini yoksa oluşturuluyor
            rm -rf $ROOTBUILDDIR/* #içeriği temizleniyor
            cd $ROOTBUILDDIR #dizinine geçiyoruz
    wget ${source}
    for f in *\ *; do mv "$f" "${f// /}"; done #isimde boşluk varsa silme işlemi yapılıyor
            dowloadfile=$(ls|head -1)
            filetype=$(file -b --extension $dowloadfile|cut -d'/' -f1)
            if [ "${filetype}" == "???" ]; then unzip  ${dowloadfile}; else tar -xvf ${dowloadfile};fi
            director=$(find ./* -maxdepth 0 -type d)
            directorname=$(basename ${director})
            if [ "${directorname}" != "${name}-${version}" ]; then mv $directorname ${name}-${version};fi
            mkdir -p $BUILDDIR&&mkdir -p $DESTDIR&&cd $SOURCEDIR
}

setup(){
        ./configure --prefix=/usr --libdir=/usr/lib64/
}

build(){
    make
}

package(){
    make install DESTDIR=$DESTDIR
    ${DESTDIR}/sbin/ldconfig -r ${DESTDIR}           # sistem guncelleniyor
}
initsetup       # initsetup fonksiyonunu çalıştırır ve kaynak dosyayı indirir
setup           # setup fonksiyonu çalışır ve derleme öncesi kaynak dosyaların ayalanması sağlanır.
build           # build fonksiyonu çalışır ve kaynak dosyaları derlenir.
package         # package fonksiyonu çalışır, yükleme öncesi ayarlamalar yapılır ve yüklenir.
```

Paket Derleme Yöntemi konusunda anlatıldığı gibi derleme işlemini yapınız.

gcc

gcc paketi, C, C++ gibi dillerde yazılmış programların çalışması için gereken kütüphaneleri içerir. Debian ortamında derlemek için **sudo apt install libmpc-dev libmpfr-dev libgmp-dev libisl-dev** paketleri kurmalıyız.

Bu dağıtım için gcc kütüphaneleri olması yeterli. Bunun için sadece libgcc adında bir paket yapmamız yeterli. Fakat siz gcc derlemek isterseniz aşağıda gcc derleme scriptini paylaşıldı.

libgcc Kurma Scripti

```
#-----------------------------------------------------------------------------------------------------------------------
#!/usr/bin/env bash
name="libgcc"
version="13.1.0"
description="libgcc"
source=""
depends=""
builddepend=""
group="sys.fs"
display=":$(ls /tmp/.X11-unix/* | sed 's#/tmp/.X11-unix/X##' | head -n 1)"
#Detect the name of the display in use
user=$(who | grep '('$display')' | awk '{print $1}')
#Detect the user using such display
ROOTBUILDDIR="/home/$user/distro/build" # Derleme konumu
BUILDDIR="/home/$user/distro/build/build-${name}-${version}" #Derleme yapılan paketin derleme konumun
DESTDIR="/home/$user/distro/rootfs" #Paketin yükleneceği sistem konumu
PACKAGEDIR=$(pwd) #paketin derleme talimatının verildiği konum
SOURCEDIR="/home/$user/distro/build/${name}-${version}" #Paketin kaynak kodlarının olduğu konum
initsetup(){
mkdir -p $ROOTBUILDDIR #derleme dizini yoksa oluşturuluyor
rm -rf $ROOTBUILDDIR/* #içeriği temizleniyor
cd $ROOTBUILDDIR #dizinine geçiyoruz
wget ${source}
for f in *\ *; do mv "$f" "${f// /}"; done #isimde boşluk varsa silme işlemi yapılıyor
dowloadfile=$(ls|head -1)
filetype=$(file -b --extension $dowloadfile|cut -d'/' -f1)
if [ "${filetype}" == "???" ]; then unzip ${dowloadfile}; else tar -xvf ${dowloadfile};fi
director=$(find ./* -maxdepth 0 -type d)
directorname=$(basename ${director})
if [ "${directorname}" != "${name}-${version}" ]; then mv $directorname ${name}-${version};fi
mkdir -p $BUILDDIR&&mkdir -p $DESTDIR&&cd $SOURCEDIR
}
setup(){
mkdir -p $SOURCEDIR
cd $SOURCEDIR
#dosya indiriliyor
wget -O libgcc.kly https://github.com/kendilinuxunuyap/kly-binary-packages/raw/refs/heads/master/libgcc/libgcc-13.1.0.kly
# tar -xf libgcc.kly
tar -xf libgcc.kly
tar -xf rootfs.tar.xz -C $BUILDDIR
}
build(){
echo ""
}

package(){
cd $BUILDDIR
cp -prfv $BUILDDIR/* ${DESTDIR}/
${DESTDIR}/sbin/ldconfig -r ${DESTDIR}
# sistem guncelleniyor
}
initsetup       # initsetup fonksiyonunu çalıştırır ve kaynak dosyayı indirir
setup   # setup fonksiyonu çalışır ve derleme öncesi kaynak dosyaların ayalanması sağlanır.
build   # build fonksiyonu çalışır ve kaynak dosyaları derlenir.
package # package fonksiyonu çalışır, yükleme öncesi ayarlamalar yapılır ve yüklenir.
```

gcc Derleme Scripti

```
#-------------------------------------------------------------------------------------------------------------------------------
#!/usr/bin/env bash
version="13.1.0"
name="gcc"
depends="glibc,gmp,mpfr,libmpc,zlib,libisl"
builddepend="flex,elfutils,curl,linux-headers"
description="DOS filesystem tools - provides mkdosfs, mkfs.msdos, mkfs.vfat"
source="https://ftp.gnu.org/gnu/gcc/gcc-${version}/gcc-${version}.tar.xz"
display=":$(ls /tmp/.X11-unix/* | sed 's#/tmp/.X11-unix/X##' | head -1)"  # Display adı
user=$(who | grep "(${display})" | awk '{print $1}')                      # Display kullanıcısı

ROOT="/home/$user/distro"
ROOTBUILDDIR="$ROOT/build"                # Derleme dizini
BUILDDIR="$ROOT/build/build-${name}-${version}" # Alt dizin
DESTDIR="$ROOT/rootfs"                     # Yükleme dizini
PACKAGEDIR=$(pwd)                          # Paket dizini
SOURCEDIR="$ROOT/build/${name}-${version}" # Kaynak dizini
initsetup(){
            mkdir -p  $ROOTBUILDDIR #derleme dizini yoksa oluşturuluyor
            rm -rf $ROOTBUILDDIR/* #içeriği temizleniyor
            cd $ROOTBUILDDIR #dizinine geçiyoruz
    wget ${source}
    for f in *\ *; do mv "$f" "${f// /}"; done #isimde boşluk varsa silme işlemi yapılıyor
            dowloadfile=$(ls|head -1)
            filetype=$(file -b --extension $dowloadfile|cut -d'/' -f1)
            if [ "${filetype}" == "???" ]; then unzip  ${dowloadfile}; else tar -xvf ${dowloadfile};fi
            director=$(find ./* -maxdepth 0 -type d)
            directorname=$(basename ${director})
            if [ "${directorname}" != "${name}-${version}" ]; then mv $directorname ${name}-${version};fi
            mkdir -p $BUILDDIR&&mkdir -p $DESTDIR&&cd $BUILDDIR
}
export CFLAGS="-O2 -s"
export CXXFLAGS="-O2 -s"
unset LDFLAGS
setup(){
case $(uname -m) in
  x86_64)
    sed -i.orig '/m64=/s/lib64/lib/' gcc/config/i386/t-linux64
  ;;
esac
    cd $SOURCEDIR
    mkdir build
    cd build
    ../configure --prefix=/usr --libexecdir=/usr/libexec --mandir=/usr/share/man --infodir=/usr/share/info  \
    --enable-languages=c,c++ --with-linker-hash-style=gnu --with-system-zlib --enable-__cxa_atexit --enable-cet=auto \
    --enable-checking=release --enable-clocale=gnu --enable-default-pie --enable-default-ssp \
    --enable-gnu-indirect-function --enable-gnu-unique-object --enable-libstdcxx-backtrace --enable-link-serialization=1 \
    --enable-linker-build-id --enable-lto --disable-multilib --enable-plugin --enable-shared --enable-threads=posix \
    --disable-libssp --disable-libstdcxx-pch --disable-werror --without-zstd --disable-nls      --libdir=/usr/lib64 \
    --target=x86_64-pc-linux-gnu
}
build(){
        cd $SOURCEDIR/build
        make
}
package(){
        cd $SOURCEDIR/build
        make install DESTDIR=${DESTDIR}
        mkdir -p ${DESTDIR}/usr/lib64/
        ln -s gcc ${DESTDIR}/usr/bin/cc
        ln -s g++ ${DESTDIR}/usr/bin/cxx
        cd $DESTDIR
        while read -rd '' file; do
        case "$(file -Sib "$file")" in
            application/x-executable\;*)     # Binaries
                strip "$file" ;;
            application/x-pie-executable\;*) # Relocatable binaries
                strip "$file" ;;
        esac
    done< <(find "./" -type f -iname "*" -print0)
}
initsetup       # initsetup fonksiyonunu çalıştırır ve kaynak dosyayı indirir
setup           # setup fonksiyonu çalışır ve derleme öncesi kaynak dosyaların ayalanması sağlanır.
build           # build fonksiyonu çalışır ve kaynak dosyaları derlenir.
package         # package fonksiyonu çalışır, yükleme öncesi ayarlamalar yapılır ve yüklenir.
```

Paket Derleme Yöntemi konusunda anlatıldığı gibi derleme işlemini yapınız.

libcap-ng

libcap-ng paketi, yetki yönetimi ve güvenlik uygulamaları için kullanılan bir kütüphanedir.

Derleme

```
#-----------------------------------------------------------------------------------------------------------------------
#!/usr/bin/env bash
version="0.8.3"
name="libcap-ng"
depends="glibc,acl,openssl,libtool"
description="shell ve network copy"
source="https://github.com/stevegrubb/libcap-ng/archive/refs/tags/v${version}.zip"
display=":$(ls /tmp/.X11-unix/* | sed 's#/tmp/.X11-unix/X##' | head -1)"  # Display adı
user=$(who | grep "(${display})" | awk '{print $1}')                      # Display kullanıcısı

ROOT="/home/$user/distro"
ROOTBUILDDIR="$ROOT/build"                # Derleme dizini
BUILDDIR="$ROOT/build/build-${name}-${version}" # Alt dizin
DESTDIR="$ROOT/rootfs"                     # Yükleme dizini
PACKAGEDIR=$(pwd)                          # Paket dizini
SOURCEDIR="$ROOT/build/${name}-${version}" # Kaynak dizini


initsetup(){
            mkdir -p  $ROOTBUILDDIR #derleme dizini yoksa oluşturuluyor
            rm -rf $ROOTBUILDDIR/* #içeriği temizleniyor
            cd $ROOTBUILDDIR #dizinine geçiyoruz
    wget ${source}
    for f in *\ *; do mv "$f" "${f// /}"; done #isimde boşluk varsa silme işlemi yapılıyor
            dowloadfile=$(ls|head -1)
            filetype=$(file -b --extension $dowloadfile|cut -d'/' -f1)
            if [ "${filetype}" == "???" ]; then unzip  ${dowloadfile}; else tar -xvf ${dowloadfile};fi
            director=$(find ./* -maxdepth 0 -type d)
            directorname=$(basename ${director})
            if [ "${directorname}" != "${name}-${version}" ]; then mv $directorname ${name}-${version};fi
            mkdir -p $BUILDDIR&&mkdir -p $DESTDIR&&cd $SOURCEDIR
}

setup()
{
./autogen.sh
./configure --prefix=/usr \
  --libdir=/lib64 \
 --with-python \
        --with-python3

}
build()
{
        make
}
package()
{
        make install DESTDIR=$DESTDIR
        ${DESTDIR}/sbin/ldconfig -r ${DESTDIR}           # sistem guncelleniyor
}
initsetup       # initsetup fonksiyonunu çalıştırır ve kaynak dosyayı indirir
setup           # setup fonksiyonu çalışır ve derleme öncesi kaynak dosyaların ayalanması sağlanır.
build           # build fonksiyonu çalışır ve kaynak dosyaları derlenir.
package         # package fonksiyonu çalışır, yükleme öncesi ayarlamalar yapılır ve yüklenir.
```

Paket Derleme Yöntemi konusunda anlatıldığı gibi derleme işlemini yapınız.

gzip

gzip(GNU zip) dosyaları sıkıştırmak için kullanılan bir yazılımdır.

Derleme

```
#-----------------------------------------------------------------------------------------------------------------------
#!/usr/bin/env bash
version="1.13"
name="gzip"
depends=""
description="Standard GNU compressor"
source="https://ftp.gnu.org/gnu/gzip/${name}-${version}.zip"
display=":$(ls /tmp/.X11-unix/* | sed 's#/tmp/.X11-unix/X##' | head -1)"  # Display adı
user=$(who | grep "(${display})" | awk '{print $1}')                      # Display kullanıcısı

ROOT="/home/$user/distro"
ROOTBUILDDIR="$ROOT/build"                # Derleme dizini
BUILDDIR="$ROOT/build/build-${name}-${version}" # Alt dizin
DESTDIR="$ROOT/rootfs"                     # Yükleme dizini
PACKAGEDIR=$(pwd)                          # Paket dizini
SOURCEDIR="$ROOT/build/${name}-${version}" # Kaynak dizini


initsetup(){
            mkdir -p  $ROOTBUILDDIR #derleme dizini yoksa oluşturuluyor
            rm -rf $ROOTBUILDDIR/* #içeriği temizleniyor
            cd $ROOTBUILDDIR #dizinine geçiyoruz
    wget ${source}
    for f in *\ *; do mv "$f" "${f// /}"; done #isimde boşluk varsa silme işlemi yapılıyor
            dowloadfile=$(ls|head -1)
            filetype=$(file -b --extension $dowloadfile|cut -d'/' -f1)
            if [ "${filetype}" == "???" ]; then unzip  ${dowloadfile}; else tar -xvf ${dowloadfile};fi
            director=$(find ./* -maxdepth 0 -type d)
            directorname=$(basename ${director})
            if [ "${directorname}" != "${name}-${version}" ]; then mv $directorname ${name}-${version};fi
            mkdir -p $BUILDDIR&&mkdir -p $DESTDIR&&cd $SOURCEDIR
}

setup()
{
        export DEFS="NO_ASM"
        ./autoreconf -fvi
        ./configure --prefix=/usr \
        --libdir=/usr/lib64/
}
build()
{
        make
}
package()
{
        make install DESTDIR=$DESTDIR
        ${DESTDIR}/sbin/ldconfig -r ${DESTDIR}          # sistem guncelleniyor
}
initsetup       # initsetup fonksiyonunu çalıştırır ve kaynak dosyayı indirir
setup           # setup fonksiyonu çalışır ve derleme öncesi kaynak dosyaların ayalanması sağlanır.
build           # build fonksiyonu çalışır ve kaynak dosyaları derlenir.
package         # package fonksiyonu çalışır, yükleme öncesi ayarlamalar yapılır ve yüklenir.
```

Paket Derleme Yöntemi konusunda anlatıldığı gibi derleme işlemini yapınız.

xz-utils

xz-utils, dosyaları sıkıştırmak ve açmak için kullanılan bir yazılım paketidir.

Derleme

```
#-----------------------------------------------------------------------------------------------------------------------
#!/usr/bin/env bash
name="xz-utils"
version="5.4.5"
description="lzma compression utilities"
source="https://tukaani.org/xz/xz-${version}.tar.gz"
md5sums="66f82a9fa24623f5ea8a9ee6b4f808e2"
display=":$(ls /tmp/.X11-unix/* | sed 's#/tmp/.X11-unix/X##' | head -1)"  # Display adı
user=$(who | grep "(${display})" | awk '{print $1}')                      # Display kullanıcısı

ROOT="/home/$user/distro"
ROOTBUILDDIR="$ROOT/build"                # Derleme dizini
BUILDDIR="$ROOT/build/build-${name}-${version}" # Alt dizin
DESTDIR="$ROOT/rootfs"                     # Yükleme dizini
PACKAGEDIR=$(pwd)                          # Paket dizini
SOURCEDIR="$ROOT/build/${name}-${version}" # Kaynak dizini

initsetup(){
            mkdir -p  $ROOTBUILDDIR #derleme dizini yoksa oluşturuluyor
            rm -rf $ROOTBUILDDIR/* #içeriği temizleniyor
            cd $ROOTBUILDDIR #dizinine geçiyoruz
    wget ${source}
    for f in *\ *; do mv "$f" "${f// /}"; done #isimde boşluk varsa silme işlemi yapılıyor
            dowloadfile=$(ls|head -1)
            filetype=$(file -b --extension $dowloadfile|cut -d'/' -f1)
            if [ "${filetype}" == "???" ]; then unzip  ${dowloadfile}; else tar -xvf ${dowloadfile};fi
            director=$(find ./* -maxdepth 0 -type d)
            directorname=$(basename ${director})
            if [ "${directorname}" != "${name}-${version}" ]; then mv $directorname ${name}-${version};fi
            mkdir -p $BUILDDIR&&mkdir -p $DESTDIR&&cd $SOURCEDIR
}

setup(){
    ./configure --prefix=/usr \
        --libdir=/usr/lib64 \
        --enable-static \
        --enable-shared \
        --enable-doc \
        --enable-nls
}
build(){
    make
}
package(){
    make install DESTDIR=$DESTDIR
    ${DESTDIR}/sbin/ldconfig -r ${DESTDIR}           # sistem guncelleniyor
}
initsetup       # initsetup fonksiyonunu çalıştırır ve kaynak dosyayı indirir
setup           # setup fonksiyonu çalışır ve derleme öncesi kaynak dosyaların ayalanması sağlanır.
build           # build fonksiyonu çalışır ve kaynak dosyaları derlenir.
package         # package fonksiyonu çalışır, yükleme öncesi ayarlamalar yapılır ve yüklenir.
```

Paket Derleme Yöntemi konusunda anlatıldığı gibi derleme işlemini yapınız.

zstd

Zstd (Zstandard), hızlı veri sıkıştırma sağlayan modern yöntem kullanan pakettir.

Derleme

```
#-----------------------------------------------------------------------------------------------------------------------
#!/usr/bin/env bash
version="1.5.5"
name="zstd"
depends="glibc,readline,ncurses"
description="şıkıştırma kütüphanesi"
source="https://github.com/facebook/zstd/releases/download/v1.5.5/${name}-${version}.tar.gz"
display=":$(ls /tmp/.X11-unix/* | sed 's#/tmp/.X11-unix/X##' | head -1)"  # Display adı
user=$(who | grep "(${display})" | awk '{print $1}')                      # Display kullanıcısı

ROOT="/home/$user/distro"
ROOTBUILDDIR="$ROOT/build"                # Derleme dizini
BUILDDIR="$ROOT/build/build-${name}-${version}" # Alt dizin
DESTDIR="$ROOT/rootfs"                     # Yükleme dizini
PACKAGEDIR=$(pwd)                          # Paket dizini
SOURCEDIR="$ROOT/build/${name}-${version}" # Kaynak dizini

initsetup(){
            mkdir -p  $ROOTBUILDDIR #derleme dizini yoksa oluşturuluyor
            rm -rf $ROOTBUILDDIR/* #içeriği temizleniyor
            cd $ROOTBUILDDIR #dizinine geçiyoruz
    wget ${source}
    for f in *\ *; do mv "$f" "${f// /}"; done #isimde boşluk varsa silme işlemi yapılıyor
            dowloadfile=$(ls|head -1)
            filetype=$(file -b --extension $dowloadfile|cut -d'/' -f1)
            if [ "${filetype}" == "???" ]; then unzip  ${dowloadfile}; else tar -xvf ${dowloadfile};fi
            director=$(find ./* -maxdepth 0 -type d)
            directorname=$(basename ${director})
            if [ "${directorname}" != "${name}-${version}" ]; then mv $directorname ${name}-${version};fi
            mkdir -p $BUILDDIR&&mkdir -p $DESTDIR&&cd $SOURCEDIR
}

setup()
{
        echo ""
}
build()
{
        make
}
package()
{
        make prefix=/usr install DESTDIR=$DESTDIR
        ${DESTDIR}/sbin/ldconfig -r ${DESTDIR}           # sistem guncelleniyor
}

initsetup       # initsetup fonksiyonunu çalıştırır ve kaynak dosyayı indirir
setup           # setup fonksiyonu çalışır ve derleme öncesi kaynak dosyaların ayalanması sağlanır.
build           # build fonksiyonu çalışır ve kaynak dosyaları derlenir.
package         # package fonksiyonu çalışır, yükleme öncesi ayarlamalar yapılır ve yüklenir.
```

Paket Derleme Yöntemi konusunda anlatıldığı gibi derleme işlemini yapınız.

bzip2

bzip2, dosyaları sıkıştırmak için kullanılan pakettir.

Derleme

```
#-----------------------------------------------------------------------------------------------------------------------
#!/usr/bin/env bash
version="1.0.8"
name="bzip2"
depends="glibc,readline,ncurses"
description="şıkıştırma kütüphanesi"
source="https://sourceware.org/pub/bzip2/${name}-${version}.tar.gz"
display=":$(ls /tmp/.X11-unix/* | sed 's#/tmp/.X11-unix/X##' | head -1)"  # Display adı
user=$(who | grep "(${display})" | awk '{print $1}')                      # Display kullanıcısı

ROOT="/home/$user/distro"
ROOTBUILDDIR="$ROOT/build"                # Derleme dizini
BUILDDIR="$ROOT/build/build-${name}-${version}" # Alt dizin
DESTDIR="$ROOT/rootfs"                     # Yükleme dizini
PACKAGEDIR=$(pwd)                          # Paket dizini
SOURCEDIR="$ROOT/build/${name}-${version}" # Kaynak dizini

initsetup(){
            mkdir -p  $ROOTBUILDDIR #derleme dizini yoksa oluşturuluyor
            rm -rf $ROOTBUILDDIR/* #içeriği temizleniyor
            cd $ROOTBUILDDIR #dizinine geçiyoruz
            wget ${source}
            dowloadfile=$(ls|head -1)
            filetype=$(file -b --extension $dowloadfile|cut -d'/' -f1)
            if [ "${filetype}" == "???" ]; then unzip  ${dowloadfile}; else tar -xvf ${dowloadfile};fi
            director=$(find ./* -maxdepth 0 -type d)
            directorname=$(basename ${director})
            if [ "${directorname}" != "${name}-${version}" ]; then mv $directorname ${name}-${version};fi
            mkdir -p $BUILDDIR&&mkdir -p $DESTDIR&&cd $SOURCEDIR
}
setup(){
            cp -prvf $PACKAGEDIR/files/ $SOURCEDIR
            #sed -i -e 's:\$(PREFIX)/man:\$(PREFIX)/share/man:g' -e 's:ln -s -f $(PREFIX)/bin/:ln -s :' Makefile
            sed -i -e "s:1\.0\.4:$version:" bzip2.1 bzip2.txt Makefile-libbz2_so manual.*
}
build(){
             make -f Makefile-libbz2_so all
             make all
}
package(){
            cd $SOURCEDIR
            make PREFIX="$DESTDIR/usr" install
            install -D libbz2.so.$version "$DESTDIR"/usr/lib64/libbz2.so.$version
            ln -s libbz2.so.$version "$DESTDIR"/usr/lib64/libbz2.so
            ln -s libbz2.so.$version "$DESTDIR"/usr/lib64/libbz2.so.${version%%.*}
            mkdir -p "$DESTDIR"/usr/lib64/pkgconfig/
            install -Dm644 files/bzip2.pc -t "$DESTDIR"/usr/lib64/pkgconfig
            ${DESTDIR}/sbin/ldconfig -r ${DESTDIR}           # sistem guncelleniyor
}
initsetup       # initsetup fonksiyonunu çalıştırır ve kaynak dosyayı indirir
setup           # setup fonksiyonu çalışır ve derleme öncesi kaynak dosyaların ayalanması sağlanır.
build           # build fonksiyonu çalışır ve kaynak dosyaları derlenir.
package         # package fonksiyonu çalışır, yükleme öncesi ayarlamalar yapılır ve yüklenir.
```

Ek dosyaları indirmek için tıklayınız. **Paket Derleme Yöntemi** konusunda anlatıldığı gibi derleme işlemini yapınız.

elfutils

elfutils, ELF dosyalarının için gerekli readelf, objdump, eu-strip gibi araçları içeren pakettir.

Derleme öncesi **sudo apt install libbz2-dev liblzma-dev** komutuyla paketin yüklenmesi gerekir.

Derleme

```
#-----------------------------------------------------------------------------------------------------------------------
#!/usr/bin/env bash
name="elfutils"
version="0.190"
description="Libraries/utilities to handle ELF objects (drop in replacement for libelf)"
source="https://sourceware.org/elfutils/ftp/${version}/elfutils-${version}.tar.bz2"
depends="bzip2,xz-utils,zstd,zlib"
display=":$(ls /tmp/.X11-unix/* | sed 's#/tmp/.X11-unix/X##' | head -1)"  # Display adı
user=$(who | grep "(${display})" | awk '{print $1}')                      # Display kullanıcısı

ROOT="/home/$user/distro"
ROOTBUILDDIR="$ROOT/build"                # Derleme dizini
BUILDDIR="$ROOT/build/build-${name}-${version}" # Alt dizin
DESTDIR="$ROOT/rootfs"                     # Yükleme dizini
PACKAGEDIR=$(pwd)                          # Paket dizini
SOURCEDIR="$ROOT/build/${name}-${version}" # Kaynak dizini

initsetup(){
            mkdir -p  $ROOTBUILDDIR #derleme dizini yoksa oluşturuluyor
            rm -rf $ROOTBUILDDIR/* #içeriği temizleniyor
            cd $ROOTBUILDDIR #dizinine geçiyoruz
    wget ${source}
    for f in *\ *; do mv "$f" "${f// /}"; done #isimde boşluk varsa silme işlemi yapılıyor
            dowloadfile=$(ls|head -1)
            filetype=$(file -b --extension $dowloadfile|cut -d'/' -f1)
            if [ "${filetype}" == "???" ]; then unzip  ${dowloadfile}; else tar -xvf ${dowloadfile};fi
            director=$(find ./* -maxdepth 0 -type d)
            directorname=$(basename ${director})
            if [ "${directorname}" != "${name}-${version}" ]; then mv $directorname ${name}-${version};fi
            mkdir -p $BUILDDIR&&mkdir -p $DESTDIR&&cd $SOURCEDIR
}
setup(){
    ./configure --prefix=/usr --libdir=/usr/lib64 \
                --enable-shared --disable-debuginfod --enable-libdebuginfod=dummy --disable-thread-safety \
                --disable-valgrind --disable-nls --program-prefix="eu-" --with-bzlib --with-lzma
}
build(){
    make
}
package(){
    make install DESTDIR=$DESTDIR
    ${DESTDIR}/sbin/ldconfig -r ${DESTDIR}           # sistem guncelleniyor
}
initsetup       #initsetup fonksiyonunu çalıştırır ve kaynak dosyayı indirir
setup           #setup fonksiyonu çalışır ve derleme öncesi kaynak dosyaların ayalanması sağlanır.
build           #build fonksiyonu çalışır ve kaynak dosyaları derlenir.
package         #package fonksiyonu çalışır, yükleme öncesi ayarlamalar yapılır ve yüklenir.
```

Paket Derleme Yöntemi konusunda anlatıldığı gibi derleme işlemini yapınız.

libselinux

libselinux güvenlik politikalarının uygulanmasına yardımcı olan bir kütüphane paketidir.

Derleme

Debian ortamında bu paketin derlenmesi için; **sudo apt install libpcre2-dev libsepol-dev** komutuyla paketin kurulması gerekmektedir.

```
#-----------------------------------------------------------------------------------------------------------------------
#!/usr/bin/env bash
version="3.6"
name="libselinux"
depends=""
description="lib"
source="https://github.com/SELinuxProject/selinux/releases/download/3.6/${name}-${version}.tar.gz"
display=":$(ls /tmp/.X11-unix/* | sed 's#/tmp/.X11-unix/X##' | head -1)"  # Display adı
user=$(who | grep "(${display})" | awk '{print $1}')                      # Display kullanıcısı

ROOT="/home/$user/distro"
ROOTBUILDDIR="$ROOT/build"                # Derleme dizini
BUILDDIR="$ROOT/build/build-${name}-${version}" # Alt dizin
DESTDIR="$ROOT/rootfs"                     # Yükleme dizini
PACKAGEDIR=$(pwd)                          # Paket dizini
SOURCEDIR="$ROOT/build/${name}-${version}" # Kaynak dizini

initsetup(){
        mkdir -p  $ROOTBUILDDIR #derleme dizini yoksa oluşturuluyor
        rm -rf $ROOTBUILDDIR/* #içeriği temizleniyor
        cd $ROOTBUILDDIR #dizinine geçiyoruz
        wget ${source}
        for f in *\ *; do mv "$f" "${f// /}"; done #isimde boşluk varsa silme işlemi yapılıyor
        dowloadfile=$(ls|head -1)
        filetype=$(file -b --extension $dowloadfile|cut -d'/' -f1)
        if [ "${filetype}" == "???" ]; then unzip  ${dowloadfile}; else tar -xvf ${dowloadfile};fi
        director=$(find ./* -maxdepth 0 -type d)
        directorname=$(basename ${director})
        if [ "${directorname}" != "${name}-${version}" ]; then mv $directorname ${name}-${version};fi
        mkdir -p $BUILDDIR&&mkdir -p $DESTDIR&&cd $SOURCEDIR
}

setup(){
        cp -prvf $PACKAGEDIR/files/ $SOURCEDIR
}
build(){
        patch -Np1 -i files/lfs64.patch
        make FTS_LDLIBS="-lfts"
}
package(){
        make install DESTDIR=$DESTDIR
        ${DESTDIR}/sbin/ldconfig -r ${DESTDIR}           # sistem guncelleniyor
}
initsetup       # initsetup fonksiyonunu çalıştırır ve kaynak dosyayı indirir
setup           # setup fonksiyonu çalışır ve derleme öncesi kaynak dosyaların ayalanması sağlanır.
build           # build fonksiyonu çalışır ve kaynak dosyaları derlenir.
package         # package fonksiyonu çalışır, yükleme öncesi ayarlamalar yapılır ve yüklenir.
```

Ek dosyaları indirmek için tıklayınız. **Paket Derleme Yöntemi** konusunda anlatıldığı gibi derleme işlemini yapınız.

tar

tar paketi, dosya arşivleme ve sıkıştırma işlemleri pakettir.

Derleme

```
#-----------------------------------------------------------------------------------------------------------------------
#!/usr/bin/env bash
name="tar"
version="1.35"
description="Utility used to store, backup, and transport files"
source="https://ftp.gnu.org/gnu/tar/tar-$version.tar.xz"
depends="glibc,acl"
builddepend=""
display=":$(ls /tmp/.X11-unix/* | sed 's#/tmp/.X11-unix/X##' | head -1)"  # Display adı
user=$(who | grep "(${display})" | awk '{print $1}')                      # Display kullanıcısı

ROOT="/home/$user/distro"
ROOTBUILDDIR="$ROOT/build"                # Derleme dizini
BUILDDIR="$ROOT/build/build-${name}-${version}" # Alt dizin
DESTDIR="$ROOT/rootfs"                     # Yükleme dizini
PACKAGEDIR=$(pwd)                          # Paket dizini
SOURCEDIR="$ROOT/build/${name}-${version}" # Kaynak dizini

initsetup(){
            mkdir -p  $ROOTBUILDDIR #derleme dizini yoksa oluşturuluyor
            rm -rf $ROOTBUILDDIR/* #içeriği temizleniyor
            cd $ROOTBUILDDIR #dizinine geçiyoruz
    wget ${source}
    for f in *\ *; do mv "$f" "${f// /}"; done #isimde boşluk varsa silme işlemi yapılıyor
            dowloadfile=$(ls|head -1)
            filetype=$(file -b --extension $dowloadfile|cut -d'/' -f1)
            if [ "${filetype}" == "???" ]; then unzip  ${dowloadfile}; else tar -xvf ${dowloadfile};fi
            director=$(find ./* -maxdepth 0 -type d)
            directorname=$(basename ${director})
            if [ "${directorname}" != "${name}-${version}" ]; then mv $directorname ${name}-${version};fi
            mkdir -p $BUILDDIR&&mkdir -p $DESTDIR&&cd $SOURCEDIR
}

setup(){
        export FORCE_UNSAFE_CONFIGURE=1
        ./configure --prefix=/usr \
                --libdir=/usr/lib64 \
                --sbindir=/usr/bin \
                --libexecdir=/usr/lib64/tar \
                --localstatedir=/var \
                --enable-backup-scripts
}
build(){
        make
}
package(){
        make DESTDIR=$DESTDIR install
        ${DESTDIR}/sbin/ldconfig -r ${DESTDIR}          # sistem guncelleniyor
}
initsetup       # initsetup fonksiyonunu çalıştırır ve kaynak dosyayı indirir
setup           # setup fonksiyonu çalışır ve derleme öncesi kaynak dosyaların ayalanması sağlanır.
build           # build fonksiyonu çalışır ve kaynak dosyaları derlenir.
package         # package fonksiyonu çalışır, yükleme öncesi ayarlamalar yapılır ve yüklenir.
```

Paket Derleme Yöntemi konusunda anlatıldığı gibi derleme işlemini yapınız.

zlib

zlib, veri sıkıştırma kütüphanesidir.

Derleme

```
#-----------------------------------------------------------------------------------------------------------------------
#!/usr/bin/env bash
version="1.3"
name="zlib"
depends="glibc,readline,ncurses,flex"
description="Compression library implementing the deflate compression method found in gzip and PKZIP"
source="https://github.com/madler/zlib/archive/refs/tags/v$version.tar.gz"
display=":$(ls /tmp/.X11-unix/* | sed 's#/tmp/.X11-unix/X##' | head -1)"  # Display adı
user=$(who | grep "(${display})" | awk '{print $1}')                      # Display kullanıcısı

ROOT="/home/$user/distro"
ROOTBUILDDIR="$ROOT/build"                # Derleme dizini
BUILDDIR="$ROOT/build/build-${name}-${version}" # Alt dizin
DESTDIR="$ROOT/rootfs"                     # Yükleme dizini
PACKAGEDIR=$(pwd)                          # Paket dizini
SOURCEDIR="$ROOT/build/${name}-${version}" # Kaynak dizini

initsetup(){
            mkdir -p  $ROOTBUILDDIR #derleme dizini yoksa oluşturuluyor
            rm -rf $ROOTBUILDDIR/* #içeriği temizleniyor
            cd $ROOTBUILDDIR #dizinine geçiyoruz
    wget ${source}
    for f in *\ *; do mv "$f" "${f// /}"; done #isimde boşluk varsa silme işlemi yapılıyor
            dowloadfile=$(ls|head -1)
            filetype=$(file -b --extension $dowloadfile|cut -d'/' -f1)
            if [ "${filetype}" == "???" ]; then unzip  ${dowloadfile}; else tar -xvf ${dowloadfile};fi
            director=$(find ./* -maxdepth 0 -type d)
            directorname=$(basename ${director})
            if [ "${directorname}" != "${name}-${version}" ]; then mv $directorname ${name}-${version};fi
            mkdir -p $BUILDDIR&&mkdir -p $DESTDIR&&cd $SOURCEDIR
}

setup(){
        ./configure --prefix=/usr
}
build(){
        make
}
package(){
        make install pkgconfigdir="/usr/lib64/pkgconfig" DESTDIR=$DESTDIR
        ${DESTDIR}/sbin/ldconfig -r ${DESTDIR}           # sistem guncelleniyor
}
initsetup       # initsetup fonksiyonunu çalıştırır ve kaynak dosyayı indirir
setup           # setup fonksiyonu çalışır ve derleme öncesi kaynak dosyaların ayalanması sağlanır.
build           # build fonksiyonu çalışır ve kaynak dosyaları derlenir.
package         # package fonksiyonu çalışır, yükleme öncesi ayarlamalar yapılır ve yüklenir.
```

Paket Derleme Yöntemi konusunda anlatıldığı gibi derleme işlemini yapınız.

brotli

Brotli, veri sıkıştırmak için kullanılan bir algoritmadır.

Derleme

Debian ortamında bu paketin derlenmesi için;

- **sudo apt install cmake** komutuyla paketin kurulması gerekmektedir.

```
#-----------------------------------------------------------------------------------------------------------------------
#!/usr/bin/env bash
version="1.1.0"
name="brotli"
depends="glibc,zlib"
description="Generic-purpose lossless compression algorithm"
source="https://github.com/google/brotli/archive/refs/tags/v$version.tar.gz"
display=":$(ls /tmp/.X11-unix/* | sed 's#/tmp/.X11-unix/X##' | head -1)"  # Display adı
user=$(who | grep "(${display})" | awk '{print $1}')                      # Display kullanıcısı

ROOT="/home/$user/distro"
ROOTBUILDDIR="$ROOT/build"                # Derleme dizini
BUILDDIR="$ROOT/build/build-${name}-${version}" # Alt dizin
DESTDIR="$ROOT/rootfs"                     # Yükleme dizini
PACKAGEDIR=$(pwd)                          # Paket dizini
SOURCEDIR="$ROOT/build/${name}-${version}" # Kaynak dizini

initsetup(){
            mkdir -p  $ROOTBUILDDIR #derleme dizini yoksa oluşturuluyor
            rm -rf $ROOTBUILDDIR/* #içeriği temizleniyor
            cd $ROOTBUILDDIR #dizinine geçiyoruz
    wget ${source}
    for f in *\ *; do mv "$f" "${f// /}"; done #isimde boşluk varsa silme işlemi yapılıyor
            dowloadfile=$(ls|head -1)
            filetype=$(file -b --extension $dowloadfile|cut -d'/' -f1)
            if [ "${filetype}" == "???" ]; then unzip  ${dowloadfile}; else tar -xvf ${dowloadfile};fi
            director=$(find ./* -maxdepth 0 -type d)
            directorname=$(basename ${director})
            if [ "${directorname}" != "${name}-${version}" ]; then mv $directorname ${name}-${version};fi
            mkdir -p $BUILDDIR&&mkdir -p $DESTDIR&&cd $SOURCEDIR
}

setup(){
    cmake \
        -DCMAKE_INSTALL_PREFIX=/usr \
        -DBUILD_SHARED_LIBS=True
}
build(){
        make
}
package(){
        make install DESTDIR=$DESTDIR
        mkdir -p $DESTDIR/lib
        mkdir -p $DESTDIR/lib/pkgconfig
        cp -prfv $DESTDIR/usr/lib/x86_64-linux-gnu/* $DESTDIR/lib/
        ${DESTDIR}/sbin/ldconfig -r ${DESTDIR}           # sistem guncelleniyor
}
initsetup       # initsetup fonksiyonunu çalıştırır ve kaynak dosyayı indirir
setup           # setup fonksiyonu çalışır ve derleme öncesi kaynak dosyaların ayalanması sağlanır.
build           # build fonksiyonu çalışır ve kaynak dosyaları derlenir.
package         # package fonksiyonu çalışır, yükleme öncesi ayarlamalar yapılır ve yüklenir.
```

Paket Derleme Yöntemi konusunda anlatıldığı gibi derleme işlemini yapınız.

curl

Curl, internet üzerinden veri transferi yapan bir araçtır.

Derleme

```
#-----------------------------------------------------------------------------------------------------------------------
#!/usr/bin/env bash
version="8.4.0"
name="curl"
depends="glibc,acl,openssl"
description="shell ve network copy"
source="https://curl.se/download/${name}-${version}.tar.xz"
display=":$(ls /tmp/.X11-unix/* | sed 's#/tmp/.X11-unix/X##' | head -1)"  # Display adı
user=$(who | grep "(${display})" | awk '{print $1}')                      # Display kullanıcısı

ROOT="/home/$user/distro"
ROOTBUILDDIR="$ROOT/build"                # Derleme dizini
BUILDDIR="$ROOT/build/build-${name}-${version}" # Alt dizin
DESTDIR="$ROOT/rootfs"                     # Yükleme dizini
PACKAGEDIR=$(pwd)                          # Paket dizini
SOURCEDIR="$ROOT/build/${name}-${version}" # Kaynak dizini

initsetup(){
                mkdir -p  $ROOTBUILDDIR #derleme dizini yoksa oluşturuluyor
                rm -rf $ROOTBUILDDIR/* #içeriği temizleniyor
                cd $ROOTBUILDDIR #dizinine geçiyoruz
                wget ${source}
                dowloadfile=$(ls|head -1)
                filetype=$(file -b --extension $dowloadfile|cut -d'/' -f1)
                if [ "${filetype}" == "???" ]; then unzip  ${dowloadfile}; else tar -xvf ${dowloadfile};fi
                director=$(find ./* -maxdepth 0 -type d)
                directorname=$(basename ${director})
                if [ "${directorname}" != "${name}-${version}" ]; then mv $directorname ${name}-${version};fi
                mkdir -p $BUILDDIR&&mkdir -p $DESTDIR&&cd $SOURCEDIR
}
setup(){
            opts=(
            --prefix=/usr --libdir=/usr/lib64 --disable-ldap --disable-ldaps --disable-versioned-symbols
            --enable-doh --enable-ftp --enable-ipv6 --with-ca-path=/etc/ssl/certs --with-ca-bundle=/etc/ssl/cert.pem
            --enable-threaded-resolverl --enable-websockets --without-libidn2 --without-libpsl --without-nghttp2)
              #--without-zstd --without-zlib --without-brotli --without-libssh)
            ./configure ${opts[@]} --with-openssl
}
build(){
            make
}
package(){
                make install DESTDIR=$DESTDIR
            cd $DESTDIR
            for ver in 3 4.0.0 4.1.0 4.2.0 4.3.0 4.4.0 4.5.0 4.6.0 4.7.0; do
            ln -s $DESTDIR/lib/libcurl.so.4.8.0 $DESTDIR/lib/libcurl.so.${ver}
                done
                ${DESTDIR}/sbin/ldconfig -r ${DESTDIR}           # sistem guncelleniyor
}
initsetup       # initsetup fonksiyonunu çalıştırır ve kaynak dosyayı indirir
setup           # setup fonksiyonu çalışır ve derleme öncesi kaynak dosyaların ayalanması sağlanır.
build           # build fonksiyonu çalışır ve kaynak dosyaları derlenir.
package         # package fonksiyonu çalışır, yükleme öncesi ayarlamalar yapılır ve yüklenir.
```

Paket Derleme Yöntemi konusunda anlatıldığı gibi derleme işlemini yapınız.

shadow

Shadow paketi, kullanıcı hesaplarının şifrelerini yönetmek için kullanılan bir yapıdır. Debian ortamında bu paketin derlenmesi için; **sudo apt install libreadline-dev libcap-dev libcap2-bin libpam0g-dev** komutları çalıştırıldıktan sonra derleme yapılmalıdır.

```
#-------------------------------------------------------------------------------------------------------------------------------
#!/usr/bin/env bash
name="shadow"
version="4.13"
description="Password and account management tool suite with support for shadow files and PAM"
source="https://github.com/shadow-maint/shadow/releases/download/$version/shadow-$version.tar.xz"
depends="pam,libxcrypt,acl,attr"
display=":$(ls /tmp/.X11-unix/* | sed 's#/tmp/.X11-unix/X##' | head -1)"  # Display adı
user=$(who | grep "(${display})" | awk '{print $1}')                      # Display kullanıcısı

ROOT="/home/$user/distro"
ROOTBUILDDIR="$ROOT/build"               # Derleme dizini
BUILDDIR="$ROOT/build/build-${name}-${version}" # Alt dizin
DESTDIR="$ROOT/rootfs"                    # Yükleme dizini
PACKAGEDIR=$(pwd)                         # Paket dizini
SOURCEDIR="$ROOT/build/${name}-${version}" # Kaynak dizini

initsetup(){
        mkdir -p  $ROOTBUILDDIR #derleme dizini yoksa oluşturuluyor
        rm -rf $ROOTBUILDDIR/* #içeriği temizleniyor
        cd $ROOTBUILDDIR #dizinine geçiyoruz
        wget ${source}
        for f in *\ *; do mv "$f" "${f// /}"; done #isimde boşluk varsa silme işlemi yapılıyor
        dowloadfile=$(ls|head -1)
        filetype=$(file -b --extension $dowloadfile|cut -d'/' -f1)
        if [ "${filetype}" == "???" ]; then unzip  ${dowloadfile}; else tar -xvf ${dowloadfile};fi
        director=$(find ./* -maxdepth 0 -type d)
        directorname=$(basename ${director})
        if [ "${directorname}" != "${name}-${version}" ]; then mv $directorname ${name}-${version};fi
        mkdir -p $BUILDDIR&&mkdir -p $DESTDIR&&cd $SOURCEDIR
}

setup(){
        tempPath="${PATH}"
        export PATH=$PATH:/usr/sbin/
        cp -prvf $PACKAGEDIR/files/ $SOURCEDIR/
        autoreconf -fiv
        ./configure --prefix=/usr --libdir=/usr/lib64 --sysconfdir=/etc --bindir=/usr/bin --sbindir=/usr/sbin \
        --disable-account-tools-setuid --without-sssd --with-fcaps --with-libpam --without-group-name-max-length \
        --with-bcrypt --with-yescrypt --without-selinux
}

build(){
    make
}
package(){
        make install DESTDIR=$DESTDIR
        mkdir -p "${DESTDIR}/etc" "${DESTDIR}/etc/default/"
        sed -i "/.*selinux.*/d" ${DESTDIR}/etc/pam.d/*
        install -vDm 600 $SOURCEDIR/files/useradd.defaults "${DESTDIR}/etc/default/useradd"
        install -vDm 600 $SOURCEDIR/files/system-auth "${DESTDIR}/etc/pam.d/system-auth"
        if [ ! -f ${DESTDIR}/etc/group ] ; then install -vDm 600 $SOURCEDIR/files/group "${DESTDIR}/etc/group"; fi
        if [ ! -f ${DESTDIR}/etc/shadow ] ; then echo "root:*::0:::::" > ${DESTDIR}/etc/shadow; fi
        chmod 600 ${DESTDIR}/etc/shadow
        chmod 644 ${DESTDIR}/etc/group
        chown root ${DESTDIR}/etc/group  ${DESTDIR}/etc/shadow
        chgrp root ${DESTDIR}/etc/group  ${DESTDIR}/etc/shadow

        if [ ! -f "${DESTDIR}/etc/passwd" ]; then echo -e "root:x:0:0:root:/root:/bin/sh">${DESTDIR}/etc/passwd; fi
        ${DESTDIR}/sbin/ldconfig -r ${DESTDIR}           # sistem guncelleniyor
        export PATH="$tempPath"
}
initsetup       # initsetup fonksiyonunu çalıştırır ve kaynak dosyayı indirir
setup           # setup fonksiyonu çalışır ve derleme öncesi kaynak dosyaların ayalanması sağlanır.
build           # build fonksiyonu çalışır ve kaynak dosyaları derlenir.
package         # package fonksiyonu çalışır, yükleme öncesi ayarlamalar yapılır ve yüklenir.
```

Ek dosyaları indirmek için tıklayınız. **Paket Derleme Yöntemi** konusunda anlatıldığı gibi derleme işlemini yapınız.

file

file, dosyaların yapılarını anlamak için kullanılan bir araçtır.

Derleme

```
#-----------------------------------------------------------------------------------------------------------------------
#!/usr/bin/env bash
name="file"
version="5.45"
description="Identify a files format by scanning binary data for patterns"
source=("http://ftp.astron.com/pub/file/file-${version}.tar.gz")
depends=""

display=":$(ls /tmp/.X11-unix/* | sed 's#/tmp/.X11-unix/X##' | head -1)"  # Display adı
user=$(who | grep "(${display})" | awk '{print $1}')                      # Display kullanıcısı

ROOT="/home/$user/distro"
ROOTBUILDDIR="$ROOT/build"                # Derleme dizini
BUILDDIR="$ROOT/build/build-${name}-${version}" # Alt dizin
DESTDIR="$ROOT/rootfs"                     # Yükleme dizini
PACKAGEDIR=$(pwd)                          # Paket dizini
SOURCEDIR="$ROOT/build/${name}-${version}" # Kaynak dizini

initsetup(){
            mkdir -p  $ROOTBUILDDIR #derleme dizini yoksa oluşturuluyor
            rm -rf $ROOTBUILDDIR/* #içeriği temizleniyor
            cd $ROOTBUILDDIR #dizinine geçiyoruz
            wget ${source}
            dowloadfile=$(ls|head -1)
            filetype=$(file -b --extension $dowloadfile|cut -d'/' -f1)
            if [ "${filetype}" == "???" ]; then unzip  ${dowloadfile}; else tar -xvf ${dowloadfile};fi
            director=$(find ./* -maxdepth 0 -type d)
            directorname=$(basename ${director})
            if [ "${directorname}" != "${name}-${version}" ]; then mv $directorname ${name}-${version};fi
            mkdir -p $BUILDDIR&&mkdir -p $DESTDIR&&cd $SOURCEDIR
}

setup(){
    opts=(
        --prefix=/usr
        --libdir=/usr/lib64
        --enable-static
        --enable-elf
        --enable-elf-core
    )
    ./configure ${opts[@]} \
        $(use_opt zlib --enable-zlib --disable-zlib) \
        $(use_opt lzma --enable-xzlib --disable-xzlib) \
        $(use_opt bzip2 --enable-bzlib --disable-bzlib) \
        $(use_opt seccomp --enable-libseccomp --disable-libseccomp)
}

build(){
    make
}

package(){
    make install DESTDIR=$DESTDIR
    ${DESTDIR}/sbin/ldconfig -r ${DESTDIR}           # sistem guncelleniyor
}
initsetup       # initsetup fonksiyonunu çalıştırır ve kaynak dosyayı indirir
setup           # setup fonksiyonu çalışır ve derleme öncesi kaynak dosyaların ayalanması sağlanır.
build           # build fonksiyonu çalışır ve kaynak dosyaları derlenir.
package         # package fonksiyonu çalışır, yükleme öncesi ayarlamalar yapılır ve yüklenir.
```

Paket Derleme Yöntemi konusunda anlatıldığı gibi derleme işlemini yapınız.

eudev

Eudev, udev'in bir çatallanmış halidir. Udev, donanımların tanınması, yönetilmesi gerekli pakettir. Debian ortamında bu paketin derlenmesi için; **sudo apt install libkmod-dev gperf** paketlerin kurulması gerekmektedir.

```
#-------------------------------------------------------------------------------------------------------------------------------
#!/usr/bin/env bash
version="3.2.14"
name="eudev"
depends="glibc,readline,ncurses,gperf"
description="modül ve sistem iletişimi sağlayan paket"
source="https://github.com/eudev-project/eudev/releases/download/v3.2.14/${name}-${version}.tar.gz"
display=":$(ls /tmp/.X11-unix/* | sed 's#/tmp/.X11-unix/X##' | head -1)"  # Display adı
user=$(who | grep "(${display})" | awk '{print $1}')                      # Display kullanıcısı

ROOT="/home/$user/distro"
ROOTBUILDDIR="$ROOT/build"               # Derleme dizini
BUILDDIR="$ROOT/build/build-${name}-${version}" # Alt dizin
DESTDIR="$ROOT/rootfs"                    # Yükleme dizini
PACKAGEDIR=$(pwd)                         # Paket dizini
SOURCEDIR="$ROOT/build/${name}-${version}" # Kaynak dizini

initsetup(){
        mkdir -p  $ROOTBUILDDIR #derleme dizini yoksa oluşturuluyor
        rm -rf $ROOTBUILDDIR/* #içeriği temizleniyor
        cd $ROOTBUILDDIR #dizinine geçiyoruz
        wget ${source}
        for f in *\ *; do mv "$f" "${f// /}"; done #isimde boşluk varsa silme işlemi yapılıyor
        dowloadfile=$(ls|head -1)
        filetype=$(file -b --extension $dowloadfile|cut -d'/' -f1)
        if [ "${filetype}" == "???" ]; then unzip  ${dowloadfile}; else tar -xvf ${dowloadfile};fi
        director=$(find ./* -maxdepth 0 -type d)
        directorname=$(basename ${director})
        if [ "${directorname}" != "${name}-${version}" ]; then mv $directorname ${name}-${version};fi
        mkdir -p $BUILDDIR&&mkdir -p $DESTDIR&&cd $SOURCEDIR
}
setup(){
        cp $PACKAGEDIR/files/eudev.hook $SOURCEDIR
        cp $PACKAGEDIR/files/eudev.init-bottom $SOURCEDIR
        cp $PACKAGEDIR/files/eudev.init-top $SOURCEDIR

        ./configure --prefix=/usr --bindir=/sbin --sbindir=/sbin --libdir=/lib64 --disable-manpages \
        --disable-static --disable-selinux --enable-modules --enable-kmod --sysconfdir=/etc --exec-prefix=/ \
        --with-rootprefix=/ --with-rootrundir=/run --with-rootlibexecdir=/lib64/udev --enable-split-usr
}
build(){
        make
}
package(){
        make install DESTDIR=$DESTDIR
        mkdir -p ${DESTDIR}/usr/share/initramfs-tools/{hooks,scripts}
        mkdir -p ${DESTDIR}/usr/share/initramfs-tools/scripts/init-{top,bottom}
        install $SOURCEDIR/eudev.hook ${DESTDIR}/usr/share/initramfs-tools/hooks/udev
    install $SOURCEDIR/eudev.init-top ${DESTDIR}/usr/share/initramfs-tools/scripts/init-top/udev
    install $SOURCEDIR/eudev.init-bottom ${DESTDIR}/usr/share/initramfs-tools/scripts/init-bottom/udev

        cd ${DESTDIR}
        mkdir -p bin
        cd bin
        ln -s ../sbin/udevadm udevadm
        ln -s ../sbin/udevd udevd
        mkdir -p  ${DESTDIR}/usr/lib64/pkgconfig/
        cd ${DESTDIR}/usr/lib64/pkgconfig/
        ln -s ../../../lib64/pkgconfig/libudev.pc libudev.pc
        ${DESTDIR}/sbin/ldconfig -r ${DESTDIR}           # sistem guncelleniyor
}

initsetup       # initsetup fonksiyonunu çalıştırır ve kaynak dosyayı indirir
setup           # setup fonksiyonu çalışır ve derleme öncesi kaynak dosyaların ayalanması sağlanır.
build           # build fonksiyonu çalışır ve kaynak dosyaları derlenir.
package         # package fonksiyonu çalışır, yükleme öncesi ayarlamalar yapılır ve yüklenir.
```

Ek dosyaları indirmek için tıklayınız. **Paket Derleme Yöntemi** konusunda anlatıldığı gibi derleme işlemini yapınız.

cpio

CPIO dosyaları arşivleme aracıdır. En sık initrd sıkıştırmasında kullanılır.

Derleme

```
#-----------------------------------------------------------------------------------------------------------------------
#!/usr/bin/env bash
name="cpio"
version="2.15"
description="A tool to copy files into or out of a cpio or tar archive"
source="https://ftp.gnu.org/gnu/cpio/cpio-$version.tar.gz"
depends="glibc"
display=":$(ls /tmp/.X11-unix/* | sed 's#/tmp/.X11-unix/X##' | head -1)"  # Display adı
user=$(who | grep "(${display})" | awk '{print $1}')                      # Display kullanıcısı

ROOT="/home/$user/distro"
ROOTBUILDDIR="$ROOT/build"                # Derleme dizini
BUILDDIR="$ROOT/build/build-${name}-${version}" # Alt dizin
DESTDIR="$ROOT/rootfs"                     # Yükleme dizini
PACKAGEDIR=$(pwd)                          # Paket dizini
SOURCEDIR="$ROOT/build/${name}-${version}" # Kaynak dizini

initsetup(){
            mkdir -p  $ROOTBUILDDIR #derleme dizini yoksa oluşturuluyor
            rm -rf $ROOTBUILDDIR/* #içeriği temizleniyor
            cd $ROOTBUILDDIR #dizinine geçiyoruz
    wget ${source}
    for f in *\ *; do mv "$f" "${f// /}"; done #isimde boşluk varsa silme işlemi yapılıyor
            dowloadfile=$(ls|head -1)
            filetype=$(file -b --extension $dowloadfile|cut -d'/' -f1)
            if [ "${filetype}" == "???" ]; then unzip  ${dowloadfile}; else tar -xvf ${dowloadfile};fi
            director=$(find ./* -maxdepth 0 -type d)
            directorname=$(basename ${director})
            if [ "${directorname}" != "${name}-${version}" ]; then mv $directorname ${name}-${version};fi
            mkdir -p $BUILDDIR&&mkdir -p $DESTDIR&&cd $SOURCEDIR
}

setup(){
    CFLAGS+=' -fcommon'
    ./configure --prefix=/usr
}
build(){
    make
}

package(){
    make install DESTDIR=$DESTDIR
    ${DESTDIR}/sbin/ldconfig -r ${DESTDIR}           # sistem guncelleniyor
}
initsetup       # initsetup fonksiyonunu çalıştırır ve kaynak dosyayı indirir
setup           # setup fonksiyonu çalışır ve derleme öncesi kaynak dosyaların ayalanması sağlanır.
build           # build fonksiyonu çalışır ve kaynak dosyaları derlenir.
package         # package fonksiyonu çalışır, yükleme öncesi ayarlamalar yapılır ve yüklenir.
```

Paket Derleme Yöntemi konusunda anlatıldığı gibi derleme işlemini yapınız.

libsepol

libsepol, SELinux'un bir parçasıdır. libsepol, güvenlik politikalarının tanımlanması ve yönetilmesi için gereken pakettir.

Derleme

Debian ortamında bu paketin derlenmesi için;

- **sudo apt install flex**

komutuyla paketin kurulması gerekmektedir.

```
#-----------------------------------------------------------------------------------------------------------------------
#!/usr/bin/env bash
version="3.6"
name="libsepol"
depends=""
description="lib"
source="https://github.com/SELinuxProject/selinux/releases/download/3.6/${name}-${version}.tar.gz"
display=":$(ls /tmp/.X11-unix/* | sed 's#/tmp/.X11-unix/X##' | head -1)"  # Display adı
user=$(who | grep "(${display})" | awk '{print $1}')                      # Display kullanıcısı

ROOT="/home/$user/distro"
ROOTBUILDDIR="$ROOT/build"                # Derleme dizini
BUILDDIR="$ROOT/build/build-${name}-${version}" # Alt dizin
DESTDIR="$ROOT/rootfs"                     # Yükleme dizini
PACKAGEDIR=$(pwd)                          # Paket dizini
SOURCEDIR="$ROOT/build/${name}-${version}" # Kaynak dizini

initsetup(){
            mkdir -p  $ROOTBUILDDIR #derleme dizini yoksa oluşturuluyor
            rm -rf $ROOTBUILDDIR/* #içeriği temizleniyor
            cd $ROOTBUILDDIR #dizinine geçiyoruz
    wget ${source}
    for f in *\ *; do mv "$f" "${f// /}"; done #isimde boşluk varsa silme işlemi yapılıyor
            dowloadfile=$(ls|head -1)
            filetype=$(file -b --extension $dowloadfile|cut -d'/' -f1)
            if [ "${filetype}" == "???" ]; then unzip  ${dowloadfile}; else tar -xvf ${dowloadfile};fi
            director=$(find ./* -maxdepth 0 -type d)
            directorname=$(basename ${director})
            if [ "${directorname}" != "${name}-${version}" ]; then mv $directorname ${name}-${version};fi
            mkdir -p $BUILDDIR&&mkdir -p $DESTDIR&&cd $SOURCEDIR
}

setup() {
        echo ""
}
build() {
        make
}
package()       {
        make install DESTDIR=$DESTDIR
        ${DESTDIR}/sbin/ldconfig -r ${DESTDIR}          # sistem guncelleniyor
}
initsetup       # initsetup fonksiyonunu çalıştırır ve kaynak dosyayı indirir
setup           # setup fonksiyonu çalışır ve derleme öncesi kaynak dosyaların ayalanması sağlanır.
build           # build fonksiyonu çalışır ve kaynak dosyaları derlenir.
package         # package fonksiyonu çalışır, yükleme öncesi ayarlamalar yapılır ve yüklenir.
```

Paket Derleme Yöntemi konusunda anlatıldığı gibi derleme işlemini yapınız.

kmod

Linux çekirdeği ile donanım arasındaki haberleşmeyi sağlayan kod parçalarını(modülleri) kerneli derlenmeden ekleme ya da çıkartmamızı sağlar.

Derleme

Debian ortamında bu paketin derlenmesi için;

- **sudo apt install libkmod-dev**

komutuyla paketin kurulması gerekmektedir.

```
#-----------------------------------------------------------------------------------------------------------------------
#!/usr/bin/env bash
name="kmod"
version="32"
description="library and tools for managing linux kernel modules"
source="https://mirrors.edge.kernel.org/pub/linux/utils/kernel/kmod/kmod-$version.tar.xz"
depends="zlib,xz-utils"
display=":$(ls /tmp/.X11-unix/* | sed 's#/tmp/.X11-unix/X##' | head -1)"  # Display adı
user=$(who | grep "(${display})" | awk '{print $1}')                      # Display kullanıcısı

ROOT="/home/$user/distro"
ROOTBUILDDIR="$ROOT/build"                # Derleme dizini
BUILDDIR="$ROOT/build/build-${name}-${version}" # Alt dizin
DESTDIR="$ROOT/rootfs"                     # Yükleme dizini
PACKAGEDIR=$(pwd)                          # Paket dizini
SOURCEDIR="$ROOT/build/${name}-${version}" # Kaynak dizini

initsetup(){
            mkdir -p  $ROOTBUILDDIR #derleme dizini yoksa oluşturuluyor
            rm -rf $ROOTBUILDDIR/* #içeriği temizleniyor
            cd $ROOTBUILDDIR #dizinine geçiyoruz
    wget ${source}
    for f in *\ *; do mv "$f" "${f// /}"; done #isimde boşluk varsa silme işlemi yapılıyor
            dowloadfile=$(ls|head -1)
            filetype=$(file -b --extension $dowloadfile|cut -d'/' -f1)
            if [ "${filetype}" == "???" ]; then unzip  ${dowloadfile}; else tar -xvf ${dowloadfile};fi
            director=$(find ./* -maxdepth 0 -type d)
            directorname=$(basename ${director})
            if [ "${directorname}" != "${name}-${version}" ]; then mv $directorname ${name}-${version};fi
            mkdir -p $BUILDDIR&&mkdir -p $DESTDIR&&cd $SOURCEDIR
}
setup(){
    touch libkmod/docs/gtk-doc.make
    ./configure --prefix=/usr --libdir=/usr/lib64/ --bindir=/bin --with-rootlibdir=/lib --with-zlib --with-openssl
}
build(){
    make
}
package(){
    make install DESTDIR=$DESTDIR
    mkdir -p ${DESTDIR}/sbin
    for i in lsmod rmmod insmod modinfo modprobe depmod; do
                ln -sf ../bin/kmod "$DESTDIR"/sbin/$i
        done
        for i in lsmod modinfo; do
                ln -s kmod "$DESTDIR"/bin/$i
        done
        ${DESTDIR}/sbin/ldconfig -r ${DESTDIR}           # sistem guncelleniyor
}
initsetup       # initsetup fonksiyonunu çalıştırır ve kaynak dosyayı indirir
setup           # setup fonksiyonu çalışır ve derleme öncesi kaynak dosyaların ayalanması sağlanır.
build           # build fonksiyonu çalışır ve kaynak dosyaları derlenir.
package         # package fonksiyonu çalışır, yükleme öncesi ayarlamalar yapılır ve yüklenir.
```

Paket Derleme Yöntemi konusunda anlatıldığı gibi derleme işlemini yapınız.

audit

Audit paketi, güvenlik denetimlerini gerçekleştirmek için tasarlanmış bir yazılımdır. Debian ortamında bu paketin derlenmesi için; **sudo apt install libaudit-dev** komutuyla paketin kurulması gerekmektedir.

```
#-----------------------------------------------------------------------------------------------------------------------
#!/usr/bin/env bash
name="audit"
version='3.1.1'
depends=""
description="servis yöneticisi"
source="https://github.com/linux-audit/audit-userspace/archive/refs/tags/v$version.tar.gz"
display=":$(ls /tmp/.X11-unix/* | sed 's#/tmp/.X11-unix/X##' | head -1)"  # Display adı
user=$(who | grep "(${display})" | awk '{print $1}')                      # Display kullanıcısı

ROOT="/home/$user/distro"
ROOTBUILDDIR="$ROOT/build"                # Derleme dizini
BUILDDIR="$ROOT/build/build-${name}-${version}" # Alt dizin
DESTDIR="$ROOT/rootfs"                     # Yükleme dizini
PACKAGEDIR=$(pwd)                          # Paket dizini
SOURCEDIR="$ROOT/build/${name}-${version}" # Kaynak dizini

initsetup(){
                mkdir -p  $ROOTBUILDDIR #derleme dizini yoksa oluşturuluyor
                rm -rf $ROOTBUILDDIR/* #içeriği temizleniyor
                cd $ROOTBUILDDIR #dizinine geçiyoruz
                wget ${source}
                for f in *\ *; do mv "$f" "${f// /}"; done #isimde boşluk varsa silme işlemi yapılıyor
                dowloadfile=$(ls|head -1)
                filetype=$(file -b --extension $dowloadfile|cut -d'/' -f1)
                if [ "${filetype}" == "???" ]; then unzip  ${dowloadfile}; else tar -xvf ${dowloadfile};fi
                director=$(find ./* -maxdepth 0 -type d)
                directorname=$(basename ${director})
                if [ "${directorname}" != "${name}-${version}" ]; then mv $directorname ${name}-${version};fi
                mkdir -p $BUILDDIR&&mkdir -p $DESTDIR&&cd $SOURCEDIR
}
setup(){
        cp -prvf $PACKAGEDIR/files/ $SOURCEDIR
        ./autogen.sh
        ./configure --prefix=/usr --sysconfdir=/etc --libdir=/usr/lib64 --disable-zos-remote --disable-listener \
        --disable-systemd --disable-gssapi-krb5 --enable-shared=audit --with-arm --with-aarch64 --without-python \
        --without-python3 --with-libcap-ng=no
}
build(){
            make
}
package(){
        make install DESTDIR=$DESTDIR
        install -Dm755 files/auditd.initd "$DESTDIR"/etc/init.d/auditd
        install -Dm755 files/auditd.confd "$DESTDIR"/etc/conf.d/auditd
        ${DESTDIR}/sbin/ldconfig -r ${DESTDIR}           # sistem guncelleniyor
}
initsetup       # initsetup fonksiyonunu çalıştırır ve kaynak dosyayı indirir
setup           # setup fonksiyonu çalışır ve derleme öncesi kaynak dosyaların ayalanması sağlanır.
build           # build fonksiyonu çalışır ve kaynak dosyaları derlenir.
package         # package fonksiyonu çalışır, yükleme öncesi ayarlamalar yapılır ve yüklenir.
```

Ek dosyaları indirmek için tıklayınız. **Paket Derleme Yöntemi** konusunda anlatıldığı gibi derleme işlemini yapınız.

libxcrypt

libxcrypt, verilerin şifrelenmesi için kullanılan bir kütüphanedir.

Derleme

```
#-----------------------------------------------------------------------------------------------------------------------
#!/usr/bin/env bash
name="libxcrypt"
version="4.4.36"
description="libxcrypt"
source=("https://github.com/besser82/libxcrypt/releases/download/v4.4.36/libxcrypt-4.4.36.tar.xz")
display=":$(ls /tmp/.X11-unix/* | sed 's#/tmp/.X11-unix/X##' | head -1)"  # Display adı
user=$(who | grep "(${display})" | awk '{print $1}')                      # Display kullanıcısı

ROOT="/home/$user/distro"
ROOTBUILDDIR="$ROOT/build"                # Derleme dizini
BUILDDIR="$ROOT/build/build-${name}-${version}" # Alt dizin
DESTDIR="$ROOT/rootfs"                     # Yükleme dizini
PACKAGEDIR=$(pwd)                          # Paket dizini
SOURCEDIR="$ROOT/build/${name}-${version}" # Kaynak dizini

initsetup(){
           mkdir -p  $ROOTBUILDDIR #derleme dizini yoksa oluşturuluyor
            rm -rf $ROOTBUILDDIR/* #içeriği temizleniyor
            cd $ROOTBUILDDIR #dizinine geçiyoruz
    wget ${source}
    for f in *\ *; do mv "$f" "${f// /}"; done #isimde boşluk varsa silme işlemi yapılıyor
            dowloadfile=$(ls|head -1)
            filetype=$(file -b --extension $dowloadfile|cut -d'/' -f1)
            if [ "${filetype}" == "???" ]; then unzip  ${dowloadfile}; else tar -xvf ${dowloadfile};fi
            director=$(find ./* -maxdepth 0 -type d)
            directorname=$(basename ${director})
            if [ "${directorname}" != "${name}-${version}" ]; then mv $directorname ${name}-${version};fi
            mkdir -p $BUILDDIR&&mkdir -p $DESTDIR&&cd $SOURCEDIR
}
setup() {
    ./configure --prefix=/usr --libdir=/usr/lib64/ --sbindir=/usr/bin
}
build() {
    make -C $SOURCEDIR
}
package() {
    make -C $SOURCEDIR install DESTDIR=$DESTDIR
    ${DESTDIR}/sbin/ldconfig -r ${DESTDIR}           # sistem guncelleniyor
}
initsetup       # initsetup fonksiyonunu çalıştırır ve kaynak dosyayı indirir
setup           # setup fonksiyonu çalışır ve derleme öncesi kaynak dosyaların ayalanması sağlanır.
build           # build fonksiyonu çalışır ve kaynak dosyaları derlenir.
package         # package fonksiyonu çalışır, yükleme öncesi ayarlamalar yapılır ve yüklenir.
```

Paket Derleme Yöntemi konusunda anlatıldığı gibi derleme işlemini yapınız.

libnsl

libnsl, ağ hizmetleri sağlamak amacıyla kullanılan pakettir.

Debian ortamında bu paketin derlenmesi için; **sudo apt install libtirpc-dev** komutuyla paketin kurulması gerekmektedir.

Derleme

```
#-----------------------------------------------------------------------------------------------------------------------
#!/usr/bin/env bash
name="libnsl"
version="2.0.0"
url="https://github.com/thkukuk/libnsl"
description="Public client interface library for NIS(YP)"
source="https://github.com/thkukuk/libnsl/releases/download/v$version/libnsl-$version.tar.xz"
depends="libtirpc"
display=":$(ls /tmp/.X11-unix/* | sed 's#/tmp/.X11-unix/X##' | head -1)"  # Display adı
user=$(who | grep "(${display})" | awk '{print $1}')                      # Display kullanıcısı

ROOT="/home/$user/distro"
ROOTBUILDDIR="$ROOT/build"                # Derleme dizini
BUILDDIR="$ROOT/build/build-${name}-${version}" # Alt dizin
DESTDIR="$ROOT/rootfs"                     # Yükleme dizini
PACKAGEDIR=$(pwd)                          # Paket dizini
SOURCEDIR="$ROOT/build/${name}-${version}" # Kaynak dizini

initsetup(){
            mkdir -p  $ROOTBUILDDIR #derleme dizini yoksa oluşturuluyor
            rm -rf $ROOTBUILDDIR/* #içeriği temizleniyor
            cd $ROOTBUILDDIR #dizinine geçiyoruz
    wget ${source}
    for f in *\ *; do mv "$f" "${f// /}"; done #isimde boşluk varsa silme işlemi yapılıyor
            dowloadfile=$(ls|head -1)
            filetype=$(file -b --extension $dowloadfile|cut -d'/' -f1)
            if [ "${filetype}" == "???" ]; then unzip  ${dowloadfile}; else tar -xvf ${dowloadfile};fi
            director=$(find ./* -maxdepth 0 -type d)
            directorname=$(basename ${director})
            if [ "${directorname}" != "${name}-${version}" ]; then mv $directorname ${name}-${version};fi
            mkdir -p $BUILDDIR&&mkdir -p $DESTDIR&&cd $SOURCEDIR
}
setup(){
    ./configure --prefix=/usr --libdir=/usr/lib64
}
build(){
    make
}
package(){
    make install DESTDIR=$DESTDIR
    ${DESTDIR}/sbin/ldconfig -r ${DESTDIR}           # sistem guncelleniyor
}
initsetup       # initsetup fonksiyonunu çalıştırır ve kaynak dosyayı indirir
setup           # setup fonksiyonu çalışır ve derleme öncesi kaynak dosyaların ayalanması sağlanır.
build           # build fonksiyonu çalışır ve kaynak dosyaları derlenir.
package         # package fonksiyonu çalışır, yükleme öncesi ayarlamalar yapılır ve yüklenir.
```

Paket Derleme Yöntemi konusunda anlatıldığı gibi derleme işlemini yapınız.

libbsd

libbsd, kod geliştirme süreçleri için gerekli pakettir.

Derleme

Derlemek için **sudo apt-get install libbsd-dev** paketi kurulmalı.

```
#-----------------------------------------------------------------------------------------------------------------------
#!/usr/bin/env bash
name="libbsd"
version="0.11.7"
description="Library to provide useful functions commonly found on BSD systems"
source="https://libbsd.freedesktop.org/releases/libbsd-$version.tar.xz"
depends=""
display=":$(ls /tmp/.X11-unix/* | sed 's#/tmp/.X11-unix/X##' | head -1)"  # Display adı
user=$(who | grep "(${display})" | awk '{print $1}')                      # Display kullanıcısı

ROOT="/home/$user/distro"
ROOTBUILDDIR="$ROOT/build"                # Derleme dizini
BUILDDIR="$ROOT/build/build-${name}-${version}" # Alt dizin
DESTDIR="$ROOT/rootfs"                     # Yükleme dizini
PACKAGEDIR=$(pwd)                          # Paket dizini
SOURCEDIR="$ROOT/build/${name}-${version}" # Kaynak dizini

initsetup(){
            mkdir -p  $ROOTBUILDDIR #derleme dizini yoksa oluşturuluyor
            rm -rf $ROOTBUILDDIR/* #içeriği temizleniyor
            cd $ROOTBUILDDIR #dizinine geçiyoruz
            wget ${source}
            for f in *\ *; do mv "$f" "${f// /}"; done #isimde boşluk varsa silme işlemi yapılıyor
            dowloadfile=$(ls|head -1)
            filetype=$(file -b --extension $dowloadfile|cut -d'/' -f1)
            if [ "${filetype}" == "???" ]; then unzip  ${dowloadfile}; else tar -xvf ${dowloadfile};fi
            director=$(find ./* -maxdepth 0 -type d)
            directorname=$(basename ${director})
            if [ "${directorname}" != "${name}-${version}" ]; then mv $directorname ${name}-${version};fi
            mkdir -p $BUILDDIR&&mkdir -p $DESTDIR&&cd $SOURCEDIR
}
setup(){
        autoreconf -fvi
        ./configure --prefix=/usr --libdir=/usr/lib64
}

build(){
        make
}

package(){
        make install DESTDIR=$DESTDIR
        ${DESTDIR}/sbin/ldconfig -r ${DESTDIR}           # sistem guncelleniyor
}

initsetup       # initsetup fonksiyonunu çalıştırır ve kaynak dosyayı indirir
setup           # setup fonksiyonu çalışır ve derleme öncesi kaynak dosyaların ayalanması sağlanır.
build           # build fonksiyonu çalışır ve kaynak dosyaları derlenir.
package         # package fonksiyonu çalışır, yükleme öncesi ayarlamalar yapılır ve yüklenir.
```

Paket Derleme Yöntemi konusunda anlatıldığı gibi derleme işlemini yapınız.

libtirpc

libtirpc, istemci ve sunucu uygulamaları arasında iletişimi kolaylaştırmak amacıyla geliştirilmiş pakettir.

Derleme

```
#-----------------------------------------------------------------------------------------------------------------------
#!/usr/bin/env bash
name="libtirpc"
version="1.3.3"
description="Transport Independent RPC library (SunRPC replacement)"
source="https://downloads.sourceforge.net/libtirpc/libtirpc-$version.tar.bz2"
depends=""
display=":$(ls /tmp/.X11-unix/* | sed 's#/tmp/.X11-unix/X##' | head -1)"  # Display adı
user=$(who | grep "(${display})" | awk '{print $1}')                      # Display kullanıcısı

ROOT="/home/$user/distro"
ROOTBUILDDIR="$ROOT/build"                # Derleme dizini
BUILDDIR="$ROOT/build/build-${name}-${version}" # Alt dizin
DESTDIR="$ROOT/rootfs"                     # Yükleme dizini
PACKAGEDIR=$(pwd)                          # Paket dizini
SOURCEDIR="$ROOT/build/${name}-${version}" # Kaynak dizini

initsetup(){
            mkdir -p  $ROOTBUILDDIR #derleme dizini yoksa oluşturuluyor
            rm -rf $ROOTBUILDDIR/* #içeriği temizleniyor
            cd $ROOTBUILDDIR #dizinine geçiyoruz
    wget ${source}
    for f in *\ *; do mv "$f" "${f// /}"; done #isimde boşluk varsa silme işlemi yapılıyor
            dowloadfile=$(ls|head -1)
            filetype=$(file -b --extension $dowloadfile|cut -d'/' -f1)
            if [ "${filetype}" == "???" ]; then unzip  ${dowloadfile}; else tar -xvf ${dowloadfile};fi
            director=$(find ./* -maxdepth 0 -type d)
            directorname=$(basename ${director})
            if [ "${directorname}" != "${name}-${version}" ]; then mv $directorname ${name}-${version};fi
            mkdir -p $BUILDDIR&&mkdir -p $DESTDIR&&cd $SOURCEDIR
}


setup(){
        ./configure --prefix=/usr \
        --libdir=/usr/lib64 \
        --sysconfdir=/etc \
        --disable-gssapi
}

build(){
    make
}

package(){
    make install DESTDIR=$DESTDIR
    ${DESTDIR}/sbin/ldconfig -r ${DESTDIR}           # sistem guncelleniyor
}
initsetup       # initsetup fonksiyonunu çalıştırır ve kaynak dosyayı indirir
setup           # setup fonksiyonu çalışır ve derleme öncesi kaynak dosyaların ayalanması sağlanır.
build           # build fonksiyonu çalışır ve kaynak dosyaları derlenir.
package         # package fonksiyonu çalışır, yükleme öncesi ayarlamalar yapılır ve yüklenir.
```

Paket Derleme Yöntemi konusunda anlatıldığı gibi derleme işlemini yapınız.

e2fsprogs

e2fsprogs, ext2, ext3 ve ext4 dosya sistemi araçlarının paketidir.

Derleme

```
#-----------------------------------------------------------------------------------------------------------------------
#!/usr/bin/env bash
version="1.47.0"
name="e2fsprogs"
depends="glibc,readline,ncurses"
description="modül ve sistem iletişimi sağlayan paket"
source="https://mirrors.edge.kernel.org/pub/linux/kernel/people/tytso/e2fsprogs/v${version}/${name}-${version}.tar.xz"
display=":$(ls /tmp/.X11-unix/* | sed 's#/tmp/.X11-unix/X##' | head -1)"  # Display adı
user=$(who | grep "(${display})" | awk '{print $1}')                      # Display kullanıcısı

ROOT="/home/$user/distro"
ROOTBUILDDIR="$ROOT/build"                # Derleme dizini
BUILDDIR="$ROOT/build/build-${name}-${version}" # Alt dizin
DESTDIR="$ROOT/rootfs"                     # Yükleme dizini
PACKAGEDIR=$(pwd)                          # Paket dizini
SOURCEDIR="$ROOT/build/${name}-${version}" # Kaynak dizini

initsetup(){
            mkdir -p  $ROOTBUILDDIR #derleme dizini yoksa oluşturuluyor
            rm -rf $ROOTBUILDDIR/* #içeriği temizleniyor
            cd $ROOTBUILDDIR #dizinine geçiyoruz
    wget ${source}
    for f in *\ *; do mv "$f" "${f// /}"; done #isimde boşluk varsa silme işlemi yapılıyor
            dowloadfile=$(ls|head -1)
            filetype=$(file -b --extension $dowloadfile|cut -d'/' -f1)
            if [ "${filetype}" == "???" ]; then unzip  ${dowloadfile}; else tar -xvf ${dowloadfile};fi
            director=$(find ./* -maxdepth 0 -type d)
            directorname=$(basename ${director})
            if [ "${directorname}" != "${name}-${version}" ]; then mv $directorname ${name}-${version};fi
            mkdir -p $BUILDDIR&&mkdir -p $DESTDIR&&cd $SOURCEDIR
}
setup(){
        ./configure --sbindir=/usr/bin --libdir=/usr/lib64/
}
build(){
        make
}
package(){
        make install DESTDIR=$DESTDIR
        rm -rf $DESTDIR/usr/share/man/man8/fsck.8
        ${DESTDIR}/sbin/ldconfig -r ${DESTDIR}          # sistem guncelleniyor
}
initsetup       # initsetup fonksiyonunu çalıştırır ve kaynak dosyayı indirir
setup           # setup fonksiyonu çalışır ve derleme öncesi kaynak dosyaların ayalanması sağlanır.
build           # build fonksiyonu çalışır ve kaynak dosyaları derlenir.
package         # package fonksiyonu çalışır, yükleme öncesi ayarlamalar yapılır ve yüklenir.
```

Paket Derleme Yöntemi konusunda anlatıldığı gibi derleme işlemini yapınız.

dosfstools

Dosfstools, FAT dosya sistemlerini oluşturmak ve yönetmek için kullanılan araçlar paketidir.

Derleme

```
#-----------------------------------------------------------------------------------------------------------------------
#!/usr/bin/env bash
version="4.2"
name="dosfstools"
depends="glibc"
description="DOS filesystem tools - provides mkdosfs, mkfs.msdos, mkfs.vfat"
source="https://github.com/dosfstools/dosfstools/archive/refs/tags/v$version.tar.gz"
display=":$(ls /tmp/.X11-unix/* | sed 's#/tmp/.X11-unix/X##' | head -1)"  # Display adı
user=$(who | grep "(${display})" | awk '{print $1}')                      # Display kullanıcısı

ROOT="/home/$user/distro"
ROOTBUILDDIR="$ROOT/build"                # Derleme dizini
BUILDDIR="$ROOT/build/build-${name}-${version}" # Alt dizin
DESTDIR="$ROOT/rootfs"                     # Yükleme dizini
PACKAGEDIR=$(pwd)                          # Paket dizini
SOURCEDIR="$ROOT/build/${name}-${version}" # Kaynak dizini

initsetup(){
            mkdir -p  $ROOTBUILDDIR #derleme dizini yoksa oluşturuluyor
            rm -rf $ROOTBUILDDIR/* #içeriği temizleniyor
            cd $ROOTBUILDDIR #dizinine geçiyoruz
    wget ${source}
    for f in *\ *; do mv "$f" "${f// /}"; done #isimde boşluk varsa silme işlemi yapılıyor
            dowloadfile=$(ls|head -1)
            filetype=$(file -b --extension $dowloadfile|cut -d'/' -f1)
            if [ "${filetype}" == "???" ]; then unzip  ${dowloadfile}; else tar -xvf ${dowloadfile};fi
            director=$(find ./* -maxdepth 0 -type d)
            directorname=$(basename ${director})
            if [ "${directorname}" != "${name}-${version}" ]; then mv $directorname ${name}-${version};fi
            mkdir -p $BUILDDIR&&mkdir -p $DESTDIR&&cd $SOURCEDIR
}

setup(){
        ./autogen.sh
        ./configure --prefix=/usr --libdir=/usr/lib64/ --enable-compat-symlinks
}
build(){
        make
}
package(){
        make install DESTDIR=$DESTDIR
        ${DESTDIR}/sbin/ldconfig -r ${DESTDIR}           # sistem guncelleniyor
}
initsetup       # initsetup fonksiyonunu çalıştırır ve kaynak dosyayı indirir
setup           # setup fonksiyonu çalışır ve derleme öncesi kaynak dosyaların ayalanması sağlanır.
build           # build fonksiyonu çalışır ve kaynak dosyaları derlenir.
package         # package fonksiyonu çalışır, yükleme öncesi ayarlamalar yapılır ve yüklenir.
```

Paket Derleme Yöntemi konusunda anlatıldığı gibi derleme işlemini yapınız.

initramfs-tools

initramfs-tools, Debian tabanlı sistemlerde kullanılan bir araçtır ve initramfs (initial RAM file system) oluşturmak için kullanılır.

Derleme

```
#-----------------------------------------------------------------------------------------------------------------------
#!/usr/bin/env bash
version="0.142"
name="initramfs-tools"
depends="glibc,readline,ncurses"
description="initramfs  generate sağlayan paket"
source="https://salsa.debian.org/kernel-team/initramfs-tools/-/archive/v$version/initramfs-tools-v$version.tar.gz"
display=":$(ls /tmp/.X11-unix/* | sed 's#/tmp/.X11-unix/X##' | head -1)"  # Display adı
user=$(who | grep "(${display})" | awk '{print $1}')                      # Display kullanıcısı

ROOT="/home/$user/distro"
ROOTBUILDDIR="$ROOT/build"                # Derleme dizini
BUILDDIR="$ROOT/build/build-${name}-${version}" # Alt dizin
DESTDIR="$ROOT/rootfs"                     # Yükleme dizini
PACKAGEDIR=$(pwd)                          # Paket dizini
SOURCEDIR="$ROOT/build/${name}-${version}" # Kaynak dizini
initsetup(){
                mkdir -p  $ROOTBUILDDIR #derleme dizini yoksa oluşturuluyor
                rm -rf $ROOTBUILDDIR/* #içeriği temizleniyor
                cd $ROOTBUILDDIR #dizinine geçiyoruz
                wget ${source}
                for f in *\ *; do mv "$f" "${f// /}"; done #isimde boşluk varsa silme işlemi yapılıyor
                dowloadfile=$(ls|head -1)
                filetype=$(file -b --extension $dowloadfile|cut -d'/' -f1)
                if [ "${filetype}" == "???" ]; then unzip  ${dowloadfile}; else tar -xvf ${dowloadfile};fi
                director=$(find ./* -maxdepth 0 -type d)
                directorname=$(basename ${director})
                if [ "${directorname}" != "${name}-${version}" ]; then mv $directorname ${name}-${version};fi
                mkdir -p $BUILDDIR&&mkdir -p $DESTDIR&&cd $SOURCEDIR
}
setup() {
            cp -prfv $PACKAGEDIR/files/* $SOURCEDIR/
            patch -Np1 < $SOURCEDIR/patches/remove-zstd.patch
            patch -Np1 < $SOURCEDIR/patches/remove-logsave.patch
            patch -Np1 < $SOURCEDIR/patches/non-debian.patch
}
build() {
echo ""
}
package()       {
            cat debian/*.install | sed "s/\t/ /g" | tr -s " " | while read line ; do
            file=$(echo $line | cut -f1 -d" ")
            target=$(echo $line | cut -f2 -d" ")
            mkdir -p ${DESTDIR}/$target
            cp -prvf $file ${DESTDIR}/$target/
            done
            # install mkinitramfs
            cp -pvf mkinitramfs ${DESTDIR}/usr/sbin/mkinitramfs
            sed -i "s/@BUSYBOX_PACKAGES@/busybox/g" ${DESTDIR}/usr/sbin/mkinitramfs
            sed -i "s/@BUSYBOX_MIN_VERSION@/1.22.0/g" ${DESTDIR}/usr/sbin/mkinitramfs
            # Remove debian stuff
            rm -rvf ${DESTDIR}/etc/kernel
            install $SOURCEDIR/zzz-busybox ${DESTDIR}/usr/share/initramfs-tools/hooks/
            install $SOURCEDIR/modules ${DESTDIR}/usr/share/initramfs-tools/
            install $SOURCEDIR/modules ${DESTDIR}/etc/initramfs-tools/

            mkdir -p ${DESTDIR}/usr/share/initramfs-tools/conf-hooks.d
            install $SOURCEDIR/conf-hooks.d/busybox ${DESTDIR}/usr/share/initramfs-tools/conf-hooks.d/
            mkdir -p ${DESTDIR}/etc/initramfs-tools/scripts
                ${DESTDIR}/sbin/ldconfig -r ${DESTDIR}          # sistem guncelleniyor
  }
initsetup       # initsetup fonksiyonunu çalıştırır ve kaynak dosyayı indirir
setup           # setup fonksiyonu çalışır ve derleme öncesi kaynak dosyaların ayalanması sağlanır.
build           # build fonksiyonu çalışır ve kaynak dosyaları derlenir.
package         # package fonksiyonu çalışır, yükleme öncesi ayarlamalar yapılır ve yüklenir.
```

Ek dosyaları indirmek için tıklayınız. **Paket Derleme Yöntemi** konusunda anlatıldığı gibi derleme işlemini yapınız.

initramfs-tools Ek Ayarları

/etc/initramfs-tools/modules

modules dosyası initrd oluşturulma ve güncelleme sırasında ek modullerin eklenmesisini gerekbilir. **initrd** açıldığında modülün yüklenmesini istiyorsak **/etc/initramfs-tools/modules** komundaki dosyayı aşağıdaki gibi düzenlemeliyiz. Bu dosya içinde **ext4**, **vfat** ve diğer yardımcı moduller eklenmiş durumdadır.

```
#------------------------------------------------------------------------------------------------------------
vfat
fat
nls_cp437
nls_ascii
nls_utf8
ext4
```

initramfs-tools Ayarları

/usr/share/initramfs-tools/hooks/ konumundaki dosyaları dikkatlice düzenlemek gerekmektedir. Dosyaları alfabetik sırayla çalıştırdığı için **busybox zzz-busybox** şeklinde ayarlanmıştır.

initrd Oluşturma/Güncelleme

Sistemin initrd.img dosyasının güncellenmesi/oluşturulması için çalıştığınız sistemde aşağıdaki komutlarla yapılabilir.

```
/usr/sbin/update-initramfs -u -k $(uname -r) #initrd günceller
```

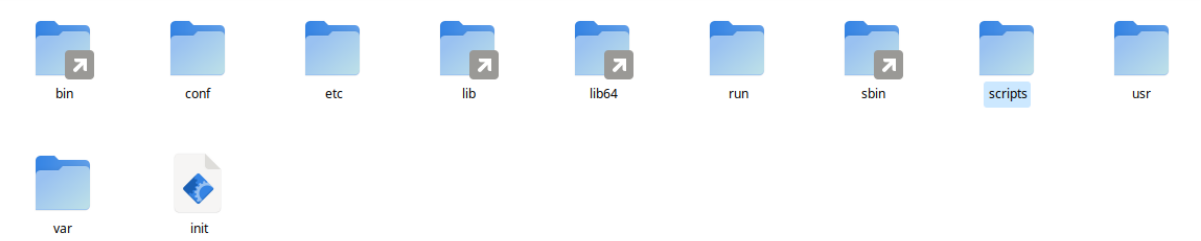
Eğer bir dizin içinde bir sisteme initrd oluşturulacaksa, yani chroot ile sisteme erişiliyorsa yukarıdaki komut yeterli olmayacaktır. chroot öncesinde sistemin **dev sys proc run** dizinlerinin bağlanılması gerekmektedir. Dikkat edilmesi gereken en önemli noktalardan biriside bu komutlar **root** yetkisiyle çalıştırılmalıdır.

```
#------------------------------------------------------------------------------------------------------------
rootfs="$HOME/distro/rootfs"
distro="$HOME/distro"
# Sisteme bağlanıyor
for dir in /dev /sys /proc /run /tmp; do
        mkdir -p $rootfs/$dir
        mount --bind /$dir $rootfs/$dir
done

### update-initrd
fname=$(basename $rootfs/boot/config*)
kversion=${fname:7}
mv $rootfs/boot/config* $rootfs/boot/config-$kversion
cp $rootfs/boot/config-$kversion $rootfs/etc/kernel-config

#sistemden ayırılıyor
chroot $rootfs update-initramfs -u -k $kversion
for dir in /dev /sys /proc /run /tmp; do
        umount -lf -R $rootfs/$dir  2>/dev/null
done
#### Copy initramfs
cp -pf $rootfs/boot/initrd.img-* $distro/iso/boot/initrd.img
```

Güncelleme ve oluşturma aşamasında **/usr/share/initramfs-tools/hooks/** konumundaki dosyarı çalıştırarak yeni initrd dosyasını oluşturacaktır. Oluşturma **/var/tmp** olacaktır. Ayrıca **/boot/config-6.6.0-amd64** gibi sistemde kullanılan kernel versiyonuyla config dosyası olmalıdır. Burada verilen **6.6.0-amd64** örnek amaçlı verilmiştir.

initrd açılma Süreci

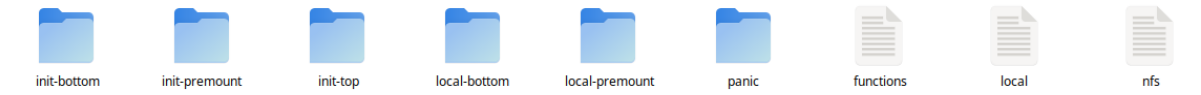
Sistemin açılması için **vmlinuz**, **initrd.img** ve **grub.cfg** dosyalarının olması yeterlidir. **initrd.img** sistemin açılma sürecini yürüten bir kernel yardımcı ön sistemidir. **initrd.img** açıldığında aşğıdaki gibi bir dizin yapısı olur. Bu dizinler içindeki **script** dizini çok önemlidir. Bu dizin içindeki scriptler belirli bir sırayla çalışarak sistemin açılması sağlanır.



initrd script İçeriği

script içerindeki dizinler aşağıdaki gibidir. Bu dizinler içinde scriptler vardır. Bu dizinlerin içeriği sırayla şöyle çalışmaktadır.

1. init-top
2. init-premount
3. init-bottom



Oluşan initrd.img dosyası sistemin açılmasını sağlayamıyorsa script açılış sürecini takip ederek sorunları çözebilirsiniz.

libxml2

libxml2, XML dosyalarını okumak, yazmak ve işlemek için kullanılan bir C kütüphanesidir.

Derleme

```
#-----------------------------------------------------------------------------------------------------------------------
#!/usr/bin/env bash
version="2.12.6"
name="libxml2"
depends="glibc,acl,openssl,libtool,icu"
builddepend="python3"
description="XML C parser and toolkit"
source="https://github.com/GNOME/libxml2/archive/refs/tags/v${version}.tar.gz"
display=":$(ls /tmp/.X11-unix/* | sed 's#/tmp/.X11-unix/X##' | head -1)"  # Display adı
user=$(who | grep "(${display})" | awk '{print $1}')                      # Display kullanıcısı

ROOT="/home/$user/distro"
ROOTBUILDDIR="$ROOT/build"                # Derleme dizini
BUILDDIR="$ROOT/build/build-${name}-${version}" # Alt dizin
DESTDIR="$ROOT/rootfs"                     # Yükleme dizini
PACKAGEDIR=$(pwd)                          # Paket dizini
SOURCEDIR="$ROOT/build/${name}-${version}" # Kaynak dizini

initsetup(){
            mkdir -p  $ROOTBUILDDIR #derleme dizini yoksa oluşturuluyor
            rm -rf $ROOTBUILDDIR/* #içeriği temizleniyor
            cd $ROOTBUILDDIR #dizinine geçiyoruz
    wget ${source}
    for f in *\ *; do mv "$f" "${f// /}"; done #isimde boşluk varsa silme işlemi yapılıyor
            dowloadfile=$(ls|head -1)
            filetype=$(file -b --extension $dowloadfile|cut -d'/' -f1)
            if [ "${filetype}" == "???" ]; then unzip  ${dowloadfile}; else tar -xvf ${dowloadfile};fi
            director=$(find ./* -maxdepth 0 -type d)
            directorname=$(basename ${director})
            if [ "${directorname}" != "${name}-${version}" ]; then mv $directorname ${name}-${version};fi
            mkdir -p $BUILDDIR&&mkdir -p $DESTDIR&&cd $SOURCEDIR
}

setup(){
./autogen.sh
./configure --prefix=/usr --libdir=/usr/lib64 --with-history --with-icu --with-legacy --with-threads
}
build(){
        make
}
package(){
        make install DESTDIR=$DESTDIR
        mkdir -p $DESTDIR/usr/lib64/python3.11
        mv $DESTDIR/usr/lib/* $DESTDIR/usr/lib64/
        ${DESTDIR}/sbin/ldconfig -r ${DESTDIR}          # sistem guncelleniyor
}
initsetup       # initsetup fonksiyonunu çalıştırır ve kaynak dosyayı indirir
setup           # setup fonksiyonu çalışır ve derleme öncesi kaynak dosyaların ayalanması sağlanır.
build           # build fonksiyonu çalışır ve kaynak dosyaları derlenir.
package         # package fonksiyonu çalışır, yükleme öncesi ayarlamalar yapılır ve yüklenir.
```

Paket Derleme Yöntemi konusunda anlatıldığı gibi derleme işlemini yapınız.

expat

Expat, XML ayrıştırma kütüphanesidir.

Derleme

```
#-------------------------------------------------------------------------------------------------------------------------
#!/usr/bin/env bash
name="expat"
version="2.6.2"
vrsn="2_6_2"
description="An XML parser library"
#source="https://github.com/libexpat/libexpat/archive/refs/tags/R_${version}.tar.gz"
source="https://github.com/libexpat/libexpat/releases/download/R_${vrsn}/expat-${version}.tar.bz2"
depends=""
builddepend=""
display=":$(ls /tmp/.X11-unix/* | sed 's#/tmp/.X11-unix/X##' | head -1)"  # Display adı
user=$(who | grep "(${display})" | awk '{print $1}')                      # Display kullanıcısı

ROOT="/home/$user/distro"
ROOTBUILDDIR="$ROOT/build"                # Derleme dizini
BUILDDIR="$ROOT/build/build-${name}-${version}" # Alt dizin
DESTDIR="$ROOT/rootfs"                     # Yükleme dizini
PACKAGEDIR=$(pwd)                          # Paket dizini
SOURCEDIR="$ROOT/build/${name}-${version}" # Kaynak dizini

initsetup(){
            mkdir -p  $ROOTBUILDDIR #derleme dizini yoksa oluşturuluyor
            rm -rf $ROOTBUILDDIR/* #içeriği temizleniyor
            cd $ROOTBUILDDIR #dizinine geçiyoruz
    wget ${source}
    for f in *\ *; do mv "$f" "${f// /}"; done #isimde boşluk varsa silme işlemi yapılıyor
            dowloadfile=$(ls|head -1)
            filetype=$(file -b --extension $dowloadfile|cut -d'/' -f1)
            if [ "${filetype}" == "???" ]; then unzip  ${dowloadfile}; else tar -xvf ${dowloadfile};fi
            director=$(find ./* -maxdepth 0 -type d)
            directorname=$(basename ${director})
            if [ "${directorname}" != "${name}-${version}" ]; then mv $directorname ${name}-${version};fi
            mkdir -p $BUILDDIR&&mkdir -p $DESTDIR&&cd $SOURCEDIR
}
setup(){
    cmake -DCMAKE_INSTALL_PREFIX=/usr \
        -DCMAKE_INSTALL_LIBDIR=lib64 \
        -DCMAKE_BUILD_TYPE=None \
        -DEXPAT_BUILD_DOCS=false \
        -W no-dev \
        -B $BUILDDIR
}
build(){
    make -C $BUILDDIR
}
package(){
    make DESTDIR="$DESTDIR" install -C $BUILDDIR
    ${DESTDIR}/sbin/ldconfig -r ${DESTDIR}           # sistem guncelleniyor
}
initsetup       # initsetup fonksiyonunu çalıştırır ve kaynak dosyayı indirir
setup           # setup fonksiyonu çalışır ve derleme öncesi kaynak dosyaların ayalanması sağlanır.
build           # build fonksiyonu çalışır ve kaynak dosyaları derlenir.
package         # package fonksiyonu çalışır, yükleme öncesi ayarlamalar yapılır ve yüklenir.
```

Paket Derleme Yöntemi konusunda anlatıldığı gibi derleme işlemini yapınız.

libmd

libmd, veri bütünlüğünü kontrol etmek için kullanılan bir kütüphanedir.

Derleme

```
#-----------------------------------------------------------------------------------------------------------------------
#!/usr/bin/env bash
name="libmd"
version="1.1.0"
description="Message Digest functions from BSD systems"
depends=""
source="https://archive.hadrons.org/software/libmd/libmd-$version.tar.xz"
display=":$(ls /tmp/.X11-unix/* | sed 's#/tmp/.X11-unix/X##' | head -1)"  # Display adı
user=$(who | grep "(${display})" | awk '{print $1}')                      # Display kullanıcısı

ROOT="/home/$user/distro"
ROOTBUILDDIR="$ROOT/build"                # Derleme dizini
BUILDDIR="$ROOT/build/build-${name}-${version}" # Alt dizin
DESTDIR="$ROOT/rootfs"                     # Yükleme dizini
PACKAGEDIR=$(pwd)                          # Paket dizini
SOURCEDIR="$ROOT/build/${name}-${version}" # Kaynak dizini

initsetup(){
            mkdir -p  $ROOTBUILDDIR #derleme dizini yoksa oluşturuluyor
            rm -rf $ROOTBUILDDIR/* #içeriği temizleniyor
            cd $ROOTBUILDDIR #dizinine geçiyoruz
    wget ${source}
    for f in *\ *; do mv "$f" "${f// /}"; done #isimde boşluk varsa silme işlemi yapılıyor
            dowloadfile=$(ls|head -1)
            filetype=$(file -b --extension $dowloadfile|cut -d'/' -f1)
            if [ "${filetype}" == "???" ]; then unzip  ${dowloadfile}; else tar -xvf ${dowloadfile};fi
            director=$(find ./* -maxdepth 0 -type d)
            directorname=$(basename ${director})
            if [ "${directorname}" != "${name}-${version}" ]; then mv $directorname ${name}-${version};fi
            mkdir -p $BUILDDIR&&mkdir -p $DESTDIR&&cd $SOURCEDIR
}
setup(){
    ./configure --prefix=/usr --libdir=/usr/lib64/
}
build(){
    make
}
package(){
    make install DESTDIR=$DESTDIR
    ${DESTDIR}/sbin/ldconfig -r ${DESTDIR}           # sistem guncelleniyor
}
initsetup       # initsetup fonksiyonunu çalıştırır ve kaynak dosyayı indirir
setup           # setup fonksiyonu çalışır ve derleme öncesi kaynak dosyaların ayalanması sağlanır.
build           # build fonksiyonu çalışır ve kaynak dosyaları derlenir.
package         # package fonksiyonu çalışır, yükleme öncesi ayarlamalar yapılır ve yüklenir.
```

Paket Derleme Yöntemi konusunda anlatıldığı gibi derleme işlemini yapınız.

libaio

Libio, C dilinde girdi/çıktı işlemlerini için kullanılan bir kütüphanedir.

Derleme

```
#-----------------------------------------------------------------------------------------------------------------------
#!/usr/bin/env bash
name="libaio"
version="0.3.113"
description="Asynchronous input/output library that uses the kernels native interface"
source="https://pagure.io/libaio/archive/libaio-$version/libaio-libaio-$version.tar.gz"
depends=""
builddepend=""
display=":$(ls /tmp/.X11-unix/* | sed 's#/tmp/.X11-unix/X##' | head -1)"  # Display adı
user=$(who | grep "(${display})" | awk '{print $1}')                      # Display kullanıcısı

ROOT="/home/$user/distro"
ROOTBUILDDIR="$ROOT/build"                # Derleme dizini
BUILDDIR="$ROOT/build/build-${name}-${version}" # Alt dizin
DESTDIR="$ROOT/rootfs"                     # Yükleme dizini
PACKAGEDIR=$(pwd)                          # Paket dizini
SOURCEDIR="$ROOT/build/${name}-${version}" # Kaynak dizini

initsetup(){
            mkdir -p  $ROOTBUILDDIR #derleme dizini yoksa oluşturuluyor
            rm -rf $ROOTBUILDDIR/* #içeriği temizleniyor
            cd $ROOTBUILDDIR #dizinine geçiyoruz
    wget ${source}
    for f in *\ *; do mv "$f" "${f// /}"; done #isimde boşluk varsa silme işlemi yapılıyor
            dowloadfile=$(ls|head -1)
            filetype=$(file -b --extension $dowloadfile|cut -d'/' -f1)
            if [ "${filetype}" == "???" ]; then unzip  ${dowloadfile}; else tar -xvf ${dowloadfile};fi
            director=$(find ./* -maxdepth 0 -type d)
            directorname=$(basename ${director})
            if [ "${directorname}" != "${name}-${version}" ]; then mv $directorname ${name}-${version};fi
            mkdir -p $BUILDDIR&&mkdir -p $DESTDIR&&cd $SOURCEDIR
}
setup (){
        echo ""
}
build(){

    make
}
package(){
    make install DESTDIR=$DESTDIR
    ${DESTDIR}/sbin/ldconfig -r ${DESTDIR}           # sistem guncelleniyor
}
initsetup       # initsetup fonksiyonunu çalıştırır ve kaynak dosyayı indirir
setup           # setup fonksiyonu çalışır ve derleme öncesi kaynak dosyaların ayalanması sağlanır.
build           # build fonksiyonu çalışır ve kaynak dosyaları derlenir.
package         # package fonksiyonu çalışır, yükleme öncesi ayarlamalar yapılır ve yüklenir.
```

Paket Derleme Yöntemi konusunda anlatıldığı gibi derleme işlemini yapınız.

lvm2

LVM2, diski yönetmek için kullanılan bir pakettir.

lvm2 Kurma Scripti

```
#-----------------------------------------------------------------------------------------------------------------------
#!/usr/bin/env bash
name="lvm2"
version="2_03_21"
description="User-land utilities for LVM2 (device-mapper) software"
source=""
depends="libaio"
builddepend=""
group="sys.fs"

display=":$(ls /tmp/.X11-unix/* | sed 's#/tmp/.X11-unix/X##' | head -n 1)"      #Detect the name of the display in use
user=$(who | grep '('$display')' | awk '{print $1}')    #Detect the user using such display
ROOTBUILDDIR="/home/$user/distro/build" # Derleme konumu
BUILDDIR="/home/$user/distro/build/build-${name}-${version}" #Derleme yapılan paketin derleme konumun
DESTDIR="/home/$user/distro/rootfs" #Paketin yükleneceği sistem konumu
PACKAGEDIR=$(pwd) #paketin derleme talimatının verildiği konum
SOURCEDIR="/home/$user/distro/build/${name}-${version}" #Paketin kaynak kodlarının olduğu konum

initsetup(){
        mkdir -p  $ROOTBUILDDIR #derleme dizini yoksa oluşturuluyor
        rm -rf $ROOTBUILDDIR/* #içeriği temizleniyor
        cd $ROOTBUILDDIR #dizinine geçiyoruz
        wget ${source}
        for f in *\ *; do mv "$f" "${f// /}"; done #isimde boşluk varsa silme işlemi yapılıyor
        dowloadfile=$(ls|head -1)
        filetype=$(file -b --extension $dowloadfile|cut -d'/' -f1)
        if [ "${filetype}" == "???" ]; then unzip  ${dowloadfile}; else tar -xvf ${dowloadfile};fi
        director=$(find ./* -maxdepth 0 -type d)
        directorname=$(basename ${director})
        if [ "${directorname}" != "${name}-${version}" ]; then mv $directorname ${name}-${version};fi
        mkdir -p $BUILDDIR&&mkdir -p $DESTDIR&&cd $SOURCEDIR
}

setup(){
        mkdir -p $SOURCEDIR
        cd $SOURCEDIR
        #dosya indiriliyor
        wget -O lvm2.kly https://github.com/kendilinuxunuyap/kly-binary-packages/raw/refs/heads/master/ext-lvm2/ext-lvm2-2_03_21.kly
        # tar -xf lvm2.kly
        tar -xf lvm2.kly
        tar -xf rootfs.tar.xz -C $BUILDDIR
}
build(){
echo ""
}
package(){
        cd $SOURCEDIR
        cp -prfv $BUILDDIR/* ${DESTDIR}/
        cd $DESTDIR/lib64
        ln -s ../usr/lib64/libdevmapper.so.1.02 libdevmapper.so.1.02.1
        ${DESTDIR}/sbin/ldconfig -r ${DESTDIR}
        # sistem guncelleniyor
}
initsetup       # initsetup fonksiyonunu çalıştırır ve kaynak dosyayı indirir
setup           # setup fonksiyonu çalışır ve derleme öncesi kaynak dosyaların ayalanması sağlanır.
build           # build fonksiyonu çalışır ve kaynak dosyaları derlenir.
package         # package fonksiyonu çalışır, yükleme öncesi ayarlamalar yapılır ve yüklenir.
```

Paket Derleme Yöntemi konusunda anlatıldığı gibi derleme işlemini yapınız.

Derleme

```
#-----------------------------------------------------------------------------------------------------------------------
#!/usr/bin/env bash
name="lvm2"
version="2_03_21"
description="User-land utilities for LVM2 (device-mapper) software"
source="https://github.com/lvmteam/lvm2/archive/refs/tags/v$version.tar.gz"
depends="libaio"
builddepend=""
display=":$(ls /tmp/.X11-unix/* | sed 's#/tmp/.X11-unix/X##' | head -1)"  # Display adı
user=$(who | grep "(${display})" | awk '{print $1}')                      # Display kullanıcısı

ROOT="/home/$user/distro"
ROOTBUILDDIR="$ROOT/build"                # Derleme dizini
BUILDDIR="$ROOT/build/build-${name}-${version}" # Alt dizin
DESTDIR="$ROOT/rootfs"                     # Yükleme dizini
PACKAGEDIR=$(pwd)                          # Paket dizini
SOURCEDIR="$ROOT/build/${name}-${version}" # Kaynak dizini

initsetup(){
            mkdir -p  $ROOTBUILDDIR #derleme dizini yoksa oluşturuluyor
            rm -rf $ROOTBUILDDIR/* #içeriği temizleniyor
            cd $ROOTBUILDDIR #dizinine geçiyoruz
    wget ${source}
    for f in *\ *; do mv "$f" "${f// /}"; done #isimde boşluk varsa silme işlemi yapılıyor
            dowloadfile=$(ls|head -1)
            filetype=$(file -b --extension $dowloadfile|cut -d'/' -f1)
            if [ "${filetype}" == "???" ]; then unzip  ${dowloadfile}; else tar -xvf ${dowloadfile};fi
            director=$(find ./* -maxdepth 0 -type d)
            directorname=$(basename ${director})
            if [ "${directorname}" != "${name}-${version}" ]; then mv $directorname ${name}-${version};fi
            mkdir -p $BUILDDIR&&mkdir -p $DESTDIR&&cd $SOURCEDIR
}
setup(){
                ./configure --prefix=/usr --libdir=/usr/lib64/  CONFIG_SHELL=/bin/bash --sbindir=/usr/bin \
                --sysconfdir=/etc --localstatedir=/var \
                --enable-cmdlib --enable-dmeventd --enable-lvmpolld --enable-pkgconfig --enable-readline \
                --enable-udev_rules --enable-udev_sync --enable-write_install --disable-systemd \
                --with-cache=internal --with-default-dm-run-dir=/run --with-default-locking-dir=/run/lock/lvm \
                --with-default-pid-dir=/run --with-default-run-dir=/run/lvm --with-thin=internal \
                --with-udev-prefix=/usr
}
build(){
    make
}
package() {
    make install DESTDIR=$DESTDIR
    ${DESTDIR}/sbin/ldconfig -r ${DESTDIR}           # sistem guncelleniyor
}

initsetup       # initsetup fonksiyonunu çalıştırır ve kaynak dosyayı indirir
setup           # setup fonksiyonu çalışır ve derleme öncesi kaynak dosyaların ayalanması sağlanır.
build           # build fonksiyonu çalışır ve kaynak dosyaları derlenir.
package         # package fonksiyonu çalışır, yükleme öncesi ayarlamalar yapılır ve yüklenir.
```

Paket Derleme Yöntemi konusunda anlatıldığı gibi derleme işlemini yapınız.

popt

popt, komut satırı parametrelerin kolayca işlenmesini sağlayan bir kütüphanedir.

Derleme

```
#-----------------------------------------------------------------------------------------------------------------------
#!/usr/bin/env bash
name="popt"
version="1.19"
description="A commandline option parser"
source="http://ftp.rpm.org/popt/releases/popt-${version%.*}.x/popt-${version}.tar.gz"
depends=""
builddepend=""
display=":$(ls /tmp/.X11-unix/* | sed 's#/tmp/.X11-unix/X##' | head -1)"  # Display adı
user=$(who | grep "(${display})" | awk '{print $1}')                      # Display kullanıcısı

ROOT="/home/$user/distro"
ROOTBUILDDIR="$ROOT/build"                # Derleme dizini
BUILDDIR="$ROOT/build/build-${name}-${version}" # Alt dizin
DESTDIR="$ROOT/rootfs"                     # Yükleme dizini
PACKAGEDIR=$(pwd)                          # Paket dizini
SOURCEDIR="$ROOT/build/${name}-${version}" # Kaynak dizini

initsetup(){
            mkdir -p  $ROOTBUILDDIR #derleme dizini yoksa oluşturuluyor
            rm -rf $ROOTBUILDDIR/* #içeriği temizleniyor
            cd $ROOTBUILDDIR #dizinine geçiyoruz
    wget ${source}
    for f in *\ *; do mv "$f" "${f// /}"; done #isimde boşluk varsa silme işlemi yapılıyor
            dowloadfile=$(ls|head -1)
            filetype=$(file -b --extension $dowloadfile|cut -d'/' -f1)
            if [ "${filetype}" == "???" ]; then unzip  ${dowloadfile}; else tar -xvf ${dowloadfile};fi
            director=$(find ./* -maxdepth 0 -type d)
            directorname=$(basename ${director})
            if [ "${directorname}" != "${name}-${version}" ]; then mv $directorname ${name}-${version};fi
            mkdir -p $BUILDDIR&&mkdir -p $DESTDIR&&cd $SOURCEDIR
}

setup(){
        CFLAGS+=" -ffat-lto-objects" ./configure --prefix=/usr
}

build(){
    make
}

package(){
    make install DESTDIR=$DESTDIR
    ${DESTDIR}/sbin/ldconfig -r ${DESTDIR}           # sistem guncelleniyor
}
initsetup       # initsetup fonksiyonunu çalıştırır ve kaynak dosyayı indirir
setup           # setup fonksiyonu çalışır ve derleme öncesi kaynak dosyaların ayalanması sağlanır.
build           # build fonksiyonu çalışır ve kaynak dosyaları derlenir.
package         # package fonksiyonu çalışır, yükleme öncesi ayarlamalar yapılır ve yüklenir.
```

Paket Derleme Yöntemi konusunda anlatıldığı gibi derleme işlemini yapınız.

icu

ICU, dil destekli uygulamalar geliştirmek metin, tarih ve saatve dil desteği sağlayan bir kütüphanedir.

Derleme

```
#-----------------------------------------------------------------------------------------------------------------------
#!/usr/bin/env bash
name="icu"
version="72-1"
description="icu International Components for Unicode"
source=("https://github.com/unicode-org/icu/releases/download/release-${version}/icu4c-74_2-src.tgz")
depends=()
display=":$(ls /tmp/.X11-unix/* | sed 's#/tmp/.X11-unix/X##' | head -1)"  # Display adı
user=$(who | grep "(${display})" | awk '{print $1}')                      # Display kullanıcısı

ROOT="/home/$user/distro"
ROOTBUILDDIR="$ROOT/build"                # Derleme dizini
BUILDDIR="$ROOT/build/build-${name}-${version}" # Alt dizin
DESTDIR="$ROOT/rootfs"                     # Yükleme dizini
PACKAGEDIR=$(pwd)                          # Paket dizini
SOURCEDIR="$ROOT/build/${name}-${version}" # Kaynak dizini

initsetup(){
            mkdir -p  $ROOTBUILDDIR #derleme dizini yoksa oluşturuluyor
            rm -rf $ROOTBUILDDIR/* #içeriği temizleniyor
            cd $ROOTBUILDDIR #dizinine geçiyoruz
    wget ${source}
    for f in *\ *; do mv "$f" "${f// /}"; done #isimde boşluk varsa silme işlemi yapılıyor
            dowloadfile=$(ls|head -1)
            filetype=$(file -b --extension $dowloadfile|cut -d'/' -f1)
            if [ "${filetype}" == "???" ]; then unzip  ${dowloadfile}; else tar -xvf ${dowloadfile};fi
            director=$(find ./* -maxdepth 0 -type d)
            directorname=$(basename ${director})
            if [ "${directorname}" != "${name}-${version}" ]; then mv $directorname ${name}-${version};fi
            mkdir -p $BUILDDIR&&mkdir -p $DESTDIR&&cd $SOURCEDIR
}

setup(){
    cd source
    ./configure --prefix=/usr \
        --libdir=/usr/lib64/
}

build(){
    make
}

package(){
    make install DESTDIR=$DESTDIR
    chmod +x "$DESTDIR"/usr/bin/icu-config
    ${DESTDIR}/sbin/ldconfig -r ${DESTDIR}           # sistem guncelleniyor
}
initsetup       # initsetup fonksiyonunu çalıştırır ve kaynak dosyayı indirir
setup           # setup fonksiyonu çalışır ve derleme öncesi kaynak dosyaların ayalanması sağlanır.
build           # build fonksiyonu çalışır ve kaynak dosyaları derlenir.
package         # package fonksiyonu çalışır, yükleme öncesi ayarlamalar yapılır ve yüklenir.
```

Paket Derleme Yöntemi konusunda anlatıldığı gibi derleme işlemini yapınız.

iproute2

iproute2, ağ yönetimi kütüphanesidir.

Derleme

```
#-----------------------------------------------------------------------------------------------------------------------
#!/usr/bin/env bash
name="iproute2"
version="6.10.0"
description="GNU regular expression matcher"
source="https://mirrors.edge.kernel.org/pub/linux/utils/net/iproute2/iproute2-6.10.0.tar.xz"
depends=""
display=":$(ls /tmp/.X11-unix/* | sed 's#/tmp/.X11-unix/X##' | head -1)"  # Display adı
user=$(who | grep "(${display})" | awk '{print $1}')                      # Display kullanıcısı

ROOT="/home/$user/distro"
ROOTBUILDDIR="$ROOT/build"                # Derleme dizini
BUILDDIR="$ROOT/build/build-${name}-${version}" # Alt dizin
DESTDIR="$ROOT/rootfs"                     # Yükleme dizini
PACKAGEDIR=$(pwd)                          # Paket dizini
SOURCEDIR="$ROOT/build/${name}-${version}" # Kaynak dizini

initsetup(){
            mkdir -p  $ROOTBUILDDIR #derleme dizini yoksa oluşturuluyor
            rm -rf $ROOTBUILDDIR/* #içeriği temizleniyor
            cd $ROOTBUILDDIR #dizinine geçiyoruz
    wget ${source}
    for f in *\ *; do mv "$f" "${f// /}"; done #isimde boşluk varsa silme işlemi yapılıyor
            dowloadfile=$(ls|head -1)
            filetype=$(file -b --extension $dowloadfile|cut -d'/' -f1)
            if [ "${filetype}" == "???" ]; then unzip  ${dowloadfile}; else tar -xvf ${dowloadfile};fi
            director=$(find ./* -maxdepth 0 -type d)
            directorname=$(basename ${director})
            if [ "${directorname}" != "${name}-${version}" ]; then mv $directorname ${name}-${version};fi
            mkdir -p $BUILDDIR&&mkdir -p $DESTDIR&&cd $SOURCEDIR
}
setup(){
        cp -prvf ${PACKAGEDIR}/files/ $SOURCEDIR/
        patch -Np1 -i $SOURCEDIR/files/0001-make-iproute2-fhs-compliant.patch
        patch -Np1 -i $SOURCEDIR/files/0002-bdb-5-3.patch
        sed -i 's/-Werror//' Makefile
        export CFLAGS+=' -ffat-lto-objects'
        ./configure
}
build(){
   make DBM_INCLUDE='/usr/include/db5.3'
}
package(){
        make DESTDIR=$DESTDIR SBINDIR="/sbin" install
        ${DESTDIR}/sbin/ldconfig -r ${DESTDIR}           # sistem guncelleniyor
}
initsetup       # initsetup fonksiyonunu çalıştırır ve kaynak dosyayı indirir
setup           # setup fonksiyonu çalışır ve derleme öncesi kaynak dosyaların ayalanması sağlanır.
build           # build fonksiyonu çalışır ve kaynak dosyaları derlenir.
package         # package fonksiyonu çalışır, yükleme öncesi ayarlamalar yapılır ve yüklenir.
```

Ek dosyaları indirmek için tıklayınız. **Paket Derleme Yöntemi** konusunda anlatıldığı gibi derleme işlemini yapınız.

net-tools

net-tools, ağ yönetimi için ifconfig, route, netstat, arp gibi araçları içerir.

Derleme

```
#-----------------------------------------------------------------------------------------------------------------------
#!/usr/bin/env bash
name="net-tools"
version="2.10"
description="GNU regular expression matcher"
source="https://sourceforge.net/projects/net-tools/files/net-tools-2.10.tar.xz"
depends=""
display=":$(ls /tmp/.X11-unix/* | sed 's#/tmp/.X11-unix/X##' | head -1)"  # Display adı
user=$(who | grep "(${display})" | awk '{print $1}')                      # Display kullanıcısı

ROOT="/home/$user/distro"
ROOTBUILDDIR="$ROOT/build"                # Derleme dizini
BUILDDIR="$ROOT/build/build-${name}-${version}" # Alt dizin
DESTDIR="$ROOT/rootfs"                     # Yükleme dizini
PACKAGEDIR=$(pwd)                          # Paket dizini
SOURCEDIR="$ROOT/build/${name}-${version}" # Kaynak dizini

initsetup(){
            mkdir -p  $ROOTBUILDDIR #derleme dizini yoksa oluşturuluyor
            rm -rf $ROOTBUILDDIR/* #içeriği temizleniyor
            cd $ROOTBUILDDIR #dizinine geçiyoruz
    wget ${source}
    for f in *\ *; do mv "$f" "${f// /}"; done #isimde boşluk varsa silme işlemi yapılıyor
            dowloadfile=$(ls|head -1)
            filetype=$(file -b --extension $dowloadfile|cut -d'/' -f1)
            if [ "${filetype}" == "???" ]; then unzip  ${dowloadfile}; else tar -xvf ${dowloadfile};fi
            director=$(find ./* -maxdepth 0 -type d)
            directorname=$(basename ${director})
            if [ "${directorname}" != "${name}-${version}" ]; then mv $directorname ${name}-${version};fi
            mkdir -p $BUILDDIR&&mkdir -p $DESTDIR&&cd $SOURCEDIR
}

setup(){
        export BINDIR='/usr/bin' SBINDIR='/usr/bin'
}

build(){
   yes "" | make
}

package(){
    make install DESTDIR=$DESTDIR
    # the following is provided by yp-tools
        rm "${DESTDIR}"/usr/bin/{nis,yp}domainname
        # hostname is provided by inetutils
        rm "${DESTDIR}"/usr/bin/{hostname,dnsdomainname,domainname}
        rm -r "${DESTDIR}"/usr/share/man/man1
        ${DESTDIR}/sbin/ldconfig -r ${DESTDIR}           # sistem guncelleniyor
}
initsetup       # initsetup fonksiyonunu çalıştırır ve kaynak dosyayı indirir
setup           # setup fonksiyonu çalışır ve derleme öncesi kaynak dosyaların ayalanması sağlanır.
build           # build fonksiyonu çalışır ve kaynak dosyaları derlenir.
package         # package fonksiyonu çalışır, yükleme öncesi ayarlamalar yapılır ve yüklenir.
```

Paket Derleme Yöntemi konusunda anlatıldığı gibi derleme işlemini yapınız.

dhcp

İstemciye uygun bir IP adresi, alt ağ maskesi, varsayılan ağ geçidi ve DNS sunucusu gibi bilgileri iletme sürecini yöneten araçlar kütüphanesidir.

Derleme

```
#-----------------------------------------------------------------------------------------------------------------------
#!/usr/bin/env bash
name="dhcp"
version="4.4.3"
description="GNU regular expression matcher"
source="https://downloads.isc.org/isc/dhcp/${version}/dhcp-${version}.tar.gz"
depends=""
display=":$(ls /tmp/.X11-unix/* | sed 's#/tmp/.X11-unix/X##' | head -1)"  # Display adı
user=$(who | grep "(${display})" | awk '{print $1}')                      # Display kullanıcısı

ROOT="/home/$user/distro"
ROOTBUILDDIR="$ROOT/build"                # Derleme dizini
BUILDDIR="$ROOT/build/build-${name}-${version}" # Alt dizin
DESTDIR="$ROOT/rootfs"                     # Yükleme dizini
PACKAGEDIR=$(pwd)                          # Paket dizini
SOURCEDIR="$ROOT/build/${name}-${version}" # Kaynak dizini

initsetup(){
            mkdir -p  $ROOTBUILDDIR #derleme dizini yoksa oluşturuluyor
            rm -rf $ROOTBUILDDIR/* #içeriği temizleniyor
            cd $ROOTBUILDDIR #dizinine geçiyoruz
    wget ${source}
    for f in *\ *; do mv "$f" "${f// /}"; done #isimde boşluk varsa silme işlemi yapılıyor
            dowloadfile=$(ls|head -1)
            filetype=$(file -b --extension $dowloadfile|cut -d'/' -f1)
            if [ "${filetype}" == "???" ]; then unzip  ${dowloadfile}; else tar -xvf ${dowloadfile};fi
            director=$(find ./* -maxdepth 0 -type d)
            directorname=$(basename ${director})
            if [ "${directorname}" != "${name}-${version}" ]; then mv $directorname ${name}-${version};fi
            mkdir -p $BUILDDIR&&mkdir -p $DESTDIR&&cd $SOURCEDIR
}
setup(){
        cp -prvf ${PACKAGEDIR}/files $SOURCEDIR/
./configure --prefix=/usr --libdir=/usr/lib64/
}
build(){
    make
}
package(){
        mkdir -p $DESTDIR/sbin/
    make install DESTDIR=$DESTDIR
    install  $SOURCEDIR/client/scripts/linux $DESTDIR/sbin/dhclient-script
    mkdir -p $DESTDIR/etc/init.d
    for level in boot default nonetwork shutdown sysinit ; do
    mkdir -p ${DESTDIR}/etc/runlevels/$level
    done
        install -Dm755  $SOURCEDIR/files/dhclient.init.d $DESTDIR/etc/init.d/dhclient
        install -Dm755  $SOURCEDIR/files/dhclient.init.d ${DESTDIR}/etc/runlevels/default/dhclient
    ${DESTDIR}/sbin/ldconfig -r ${DESTDIR}           # sistem guncelleniyor
}
initsetup       # initsetup fonksiyonunu çalıştırır ve kaynak dosyayı indirir
setup           # setup fonksiyonu çalışır ve derleme öncesi kaynak dosyaların ayalanması sağlanır.
build           # build fonksiyonu çalışır ve kaynak dosyaları derlenir.
package         # package fonksiyonu çalışır, yükleme öncesi ayarlamalar yapılır ve yüklenir.
```

Ek dosyaları indirmek için tıklayınız. **Paket Derleme Yöntemi** konusunda anlatıldığı gibi derleme işlemini yapınız.

openrc

OpenRC, sistem başlangıcını ve hizmetlerin yönetimini sağlamak amacıyla geliştirilmiş bir init sistemidir. Openrc'nin kullanımı için **Yardımcı Konular** bölümüne bakınız. Derlenmesi için; **sudo apt install libpam0g-dev** komutuyla paketlerin kurulması gerekmektedir.

```
#-----------------------------------------------------------------------------------------------------------------------
#!/usr/bin/env bash
name="openrc"
version="0.53"
description="The OpenRC init system"
source="https://github.com/OpenRC/openrc/archive/refs/tags/$version.zip"
depends=""
display=":$(ls /tmp/.X11-unix/* | sed 's#/tmp/.X11-unix/X##' | head -1)"  # Display adı
user=$(who | grep "(${display})" | awk '{print $1}')                      # Display kullanıcısı
ROOT="/home/$user/distro"
ROOTBUILDDIR="$ROOT/build"                # Derleme dizini
BUILDDIR="$ROOT/build/build-${name}-${version}" # Alt dizin
DESTDIR="$ROOT/rootfs"                     # Yükleme dizini
PACKAGEDIR=$(pwd)                          # Paket dizini
SOURCEDIR="$ROOT/build/${name}-${version}" # Kaynak dizini
initsetup(){
    mkdir -p  $ROOTBUILDDIR #derleme dizini yoksa oluşturuluyor
    rm -rf $ROOTBUILDDIR/* #içeriği temizleniyor
    cd $ROOTBUILDDIR #dizinine geçiyoruz
    wget ${source}
    for f in *\ *; do mv "$f" "${f// /}"; done #isimde boşluk varsa silme işlemi yapılıyor
    dowloadfile=$(ls|head -1)
    filetype=$(file -b --extension $dowloadfile|cut -d'/' -f1)
    if [ "${filetype}" == "???" ]; then unzip  ${dowloadfile}; else tar -xvf ${dowloadfile};fi
    director=$(find ./* -maxdepth 0 -type d)
    directorname=$(basename ${director})
    if [ "${directorname}" != "${name}-${version}" ]; then mv $directorname ${name}-${version};fi
    mkdir -p $BUILDDIR&&mkdir -p $DESTDIR&&cd $SOURCEDIR
}
setup(){
        cp -prfv $PACKAGEDIR/files $SOURCEDIR/
        cp -prfv $PACKAGEDIR/extras $SOURCEDIR/
        meson setup $BUILDDIR --sysconfdir=/etc --prefix=/ --libdir=/lib64 --includedir=/usr/include \
        -Ddefault_library=both -Dzsh-completions=true -Dbash-completions=true -Dpam=true -Dselinux=disabled -Dpkgconfig=true
}
build(){
        meson compile -C $BUILDDIR
}
package(){
        DESTDIR="$DESTDIR" meson install --no-rebuild -C $BUILDDIR
        mkdir -p "$DESTDIR/sbin"; cd "$DESTDIR/sbin"
        ln -s ../usr/sbin/openrc-init openrc-init
        ln -s ../usr/sbin/openrc-run openrc-run
        ln -s ../usr/sbin/openrc-shutdown shutdown
        cd $SOURCEDIR
        rm -f ${DESTDIR}/etc/runlevels/*/*          # disable all services
        rm ${DESTDIR}//etc/init.d/functions.sh
        ln -s ../../usr/lib/rc/sh/functions.sh ${DESTDIR}/etc/init.d/functions.sh
        mkdir -p ${DESTDIR}/usr ${DESTDIR}/sbin
        mv ${DESTDIR}/{,usr}/share          # move /share to /usr/share
        install $SOURCEDIR/files/reboot ${DESTDIR}/sbin/reboot      # reboot and poweroff script
        install $SOURCEDIR/files/poweroff ${DESTDIR}/sbin/poweroff
        ln -s openrc-shutdown ${DESTDIR}/sbin/shutdown
        mkdir -p ${DESTDIR}/usr/libexec
        install $SOURCEDIR/extras/disable-secondary-gpu.sh ${DESTDIR}/usr/libexec/disable-secondary-gpu
        install $SOURCEDIR/extras/disable-secondary-gpu.initd ${DESTDIR}/etc/init.d
        install $SOURCEDIR/extras/backlight-restore.initd ${DESTDIR}/etc/init.d
        install $SOURCEDIR/files/0modules.init.d ${DESTDIR}/etc/init.d/0modules
        for level in boot default nonetwork shutdown sysinit ; do
        mkdir -p ${DESTDIR}/etc/runlevels/$level
        done
        touch ${DESTDIR}/etc/fstab
        install $SOURCEDIR/files/0modules.init.d ${DESTDIR}/etc/init.d/0modules
        install $SOURCEDIR/files/0modules.init.d ${DESTDIR}/etc/runlevels/default/0modules
        install ${DESTDIR}/etc/init.d/hostname ${DESTDIR}/etc/runlevels/default/hostname
        cd ${DESTDIR}/etc/init.d/
        ln -s agetty agetty.tty1
        install ${DESTDIR}/etc/init.d/agetty.tty1 ${DESTDIR}/etc/runlevels/default/agetty.tty1

        install $SOURCEDIR/files/rootfspermit.init.d ${DESTDIR}/etc/init.d/rootfspermit
        install $SOURCEDIR/files/rootfspermit.local.d ${DESTDIR}/etc/local.d/rootfspermit
        cd ${DESTDIR}/etc/runlevels/default
        ln -s ../../init.d/rootfspermit rootfspermit
    ${DESTDIR}/sbin/ldconfig -r ${DESTDIR}          # sistem guncelleniyor
}
initsetup;      setup;  build;  package;
```

Ek dosyaları indirmek için; 1. Ek tıklayınız. 2.Ek tıklayınız.. **Paket Derleme Yöntemi** konusunda anlatıldığı gibi derleme işlemini yapınız.

rsync

rsync, yerel veya uzak sistemler arasında dosyaların kopyalanmasını sağlamaktır.

Debian ortamında bu paketin derlenmesi için; **sudo apt install libzstd-dev libacl1-dev libacl1 libssl-dev** komutuyla paketlerin kurulması gerekmektedir.

```
#-------------------------------------------------------------------------------------------------------------------------
#!/usr/bin/env bash
version="3.2.7"
name="rsync"
depends="glibc,acl,openssl"
description="shell ve network copy"
source="https://download.samba.org/pub/rsync/src/${name}-${version}.tar.gz"
display=":$(ls /tmp/.X11-unix/* | sed 's#/tmp/.X11-unix/X##' | head -1)"  # Display adı
user=$(who | grep "(${display})" | awk '{print $1}')                      # Display kullanıcısı

ROOT="/home/$user/distro"
ROOTBUILDDIR="$ROOT/build"                # Derleme dizini
BUILDDIR="$ROOT/build/build-${name}-${version}" # Alt dizin
DESTDIR="$ROOT/rootfs"                     # Yükleme dizini
PACKAGEDIR=$(pwd)                          # Paket dizini
SOURCEDIR="$ROOT/build/${name}-${version}" # Kaynak dizini

initsetup(){
            mkdir -p  $ROOTBUILDDIR #derleme dizini yoksa oluşturuluyor
            rm -rf $ROOTBUILDDIR/* #içeriği temizleniyor
            cd $ROOTBUILDDIR #dizinine geçiyoruz
    wget ${source}
    for f in *\ *; do mv "$f" "${f// /}"; done #isimde boşluk varsa silme işlemi yapılıyor
            dowloadfile=$(ls|head -1)
            filetype=$(file -b --extension $dowloadfile|cut -d'/' -f1)
            if [ "${filetype}" == "???" ]; then unzip  ${dowloadfile}; else tar -xvf ${dowloadfile};fi
            director=$(find ./* -maxdepth 0 -type d)
            directorname=$(basename ${director})
            if [ "${directorname}" != "${name}-${version}" ]; then mv $directorname ${name}-${version};fi
            mkdir -p $BUILDDIR&&mkdir -p $DESTDIR&&cd $SOURCEDIR
}

setup(){
        ./configure --prefix=/usr --libdir=/lib64/ --with-included-popt --with-included-zlib \
        --disable-xxhash --disable-lz4 --enable-acl-support
}
build(){
        make
}
package(){
        make install DESTDIR=$DESTDIR
        ${DESTDIR}/sbin/ldconfig -r ${DESTDIR}           # sistem guncelleniyor
}

initsetup       # initsetup fonksiyonunu çalıştırır ve kaynak dosyayı indirir
setup           # setup fonksiyonu çalışır ve derleme öncesi kaynak dosyaların ayalanması sağlanır.
build           # build fonksiyonu çalışır ve kaynak dosyaları derlenir.
package         # package fonksiyonu çalışır, yükleme öncesi ayarlamalar yapılır ve yüklenir.
```

Paket Derleme Yöntemi konusunda anlatıldığı gibi derleme işlemini yapınız.

kbd

KBD paketi,klavye ile etkileşimi yöneten kütüphanedir.

Derleme

```
#-----------------------------------------------------------------------------------------------------------------------
#!/usr/bin/env bash
name="kbd"
version="2.6.4"
description="Keytable files and keyboard utilities"
source="https://www.kernel.org/pub/linux/utils/kbd/kbd-${version}.tar.gz"
depends="pam"
makedepend="flex,autoconf,automake"
display=":$(ls /tmp/.X11-unix/* | sed 's#/tmp/.X11-unix/X##' | head -1)"  # Display adı
user=$(who | grep "(${display})" | awk '{print $1}')                      # Display kullanıcısı

ROOT="/home/$user/distro"
ROOTBUILDDIR="$ROOT/build"                # Derleme dizini
BUILDDIR="$ROOT/build/build-${name}-${version}" # Alt dizin
DESTDIR="$ROOT/rootfs"                     # Yükleme dizini
PACKAGEDIR=$(pwd)                          # Paket dizini
SOURCEDIR="$ROOT/build/${name}-${version}" # Kaynak dizini

initsetup(){
                mkdir -p  $ROOTBUILDDIR #derleme dizini yoksa oluşturuluyor
                rm -rf $ROOTBUILDDIR/* #içeriği temizleniyor
                cd $ROOTBUILDDIR #dizinine geçiyoruz
                        wget ${source}
                        for f in *\ *; do mv "$f" "${f// /}"; done #isimde boşluk varsa silme işlemi yapılıyor
                dowloadfile=$(ls|head -1)
                filetype=$(file -b --extension $dowloadfile|cut -d'/' -f1)
                if [ "${filetype}" == "???" ]; then unzip  ${dowloadfile}; else tar -xvf ${dowloadfile};fi
                director=$(find ./* -maxdepth 0 -type d)
                directorname=$(basename ${director})
                if [ "${directorname}" != "${name}-${version}" ]; then mv $directorname ${name}-${version};fi
                mkdir -p $BUILDDIR&&mkdir -p $DESTDIR&&cd $SOURCEDIR
}
setup(){
            cp -prfv $PACKAGEDIR/files $SOURCEDIR/
        autoreconf -fvi
        ./configure --prefix=/usr --sysconfdir=/etc --datadir=/usr/share/kbd --enable-optional-progs
}
build(){
        make KEYCODES_PROGS=yes RESIZECONS_PROGS=yes
}
package(){
        make DESTDIR=${DESTDIR} install
        for level in boot default nonetwork shutdown sysinit ; do
        mkdir -p ${DESTDIR}/etc/runlevels/$level
        done
        install -Dm755 $SOURCEDIR/files/loadkeys.initd "$DESTDIR"/etc/init.d/loadkeys
        install -Dm755  $SOURCEDIR/files/loadkeys.initd ${DESTDIR}/etc/runlevels/default/loadkeys

        install -Dm644 $SOURCEDIR/files/loadkeys.confd "$DESTDIR"/etc/conf.d/loadkeys
        ${DESTDIR}/sbin/ldconfig -r ${DESTDIR}           # sistem guncelleniyor
}
initsetup       # initsetup fonksiyonunu çalıştırır ve kaynak dosyayı indirir
setup           # setup fonksiyonu çalışır ve derleme öncesi kaynak dosyaların ayalanması sağlanır.
build           # build fonksiyonu çalışır ve kaynak dosyaları derlenir.
package         # package fonksiyonu çalışır, yükleme öncesi ayarlamalar yapılır ve yüklenir.
```

Ek dosyaları indirmek için tıklayınız. **Paket Derleme Yöntemi** konusunda anlatıldığı gibi derleme işlemini yapınız.

kernel

Kernel, tüm süreçleri yöneten yazılımdır. Donanım kaynaklarını yönetir, sistem çağrılarını işler ve uygulama yazılımlarının donanım ile etkileşimini sağlar.

Aşağıda nasıl derlendiği detaylıca anlatılmıştır. Derleme işlemi zaman ve tecrübe gerektirdiği için hazır derlenmiş olanı kullanacağız. Aslında debian, arch vb. dağıtımların kernelini derlemeden kullanabiliriz. Bir uyumsuzluk yaratmayacaktır. Bundan dolayı kendi derlediğimiz(basitdagitim) kernelini indirip kendi sistemimize yükleyen bir işlem yapacağız. Fakat derlemek isterseniz Derleme başlığı altında paylaşılan scripti kullanabilirsiniz. Kerneli hazırladığımız sisteme kurmak için aşağıda script verilmiştir.

Kernel Kurma Scripti

```
#-----------------------------------------------------------------------------------------------------------------------
#!/usr/bin/env bash
version="6.10.6"
name="linux-image"
depends=""
description="temel dağıtım kernel dosyası ve moduller"
source=""
groups="sys.base"

display=":$(ls /tmp/.X11-unix/* | sed 's#/tmp/.X11-unix/X##' | head -n 1)"      #Detect the name of the display in use
user=$(who | grep '('$display')' | awk '{print $1}')    #Detect the user using such display
ROOTBUILDDIR="/home/$user/distro/build" # Derleme konumu
BUILDDIR="/home/$user/distro/build/build-${name}-${version}" #Derleme yapılan paketin derleme konumun
DESTDIR="/home/$user/distro/rootfs" #Paketin yükleneceği sistem konumu
PACKAGEDIR=$(pwd) #paketin derleme talimatının verildiği konum
SOURCEDIR="/home/$user/distro/build/${name}-${version}" #Paketin kaynak kodlarının olduğu konum

initsetup(){
            mkdir -p  $ROOTBUILDDIR #derleme dizini yoksa oluşturuluyor
            rm -rf $ROOTBUILDDIR/* #içeriği temizleniyor
            cd $ROOTBUILDDIR #dizinine geçiyoruz
    wget ${source}
    for f in *\ *; do mv "$f" "${f// /}"; done #isimde boşluk varsa silme işlemi yapılıyor
            dowloadfile=$(ls|head -1)
            filetype=$(file -b --extension $dowloadfile|cut -d'/' -f1)
            if [ "${filetype}" == "???" ]; then unzip  ${dowloadfile}; else tar -xvf ${dowloadfile};fi
            director=$(find ./* -maxdepth 0 -type d)
            directorname=$(basename ${director})
            if [ "${directorname}" != "${name}-${version}" ]; then mv $directorname ${name}-${version};fi
            mkdir -p $BUILDDIR&&mkdir -p $DESTDIR&&cd $SOURCEDIR
}

setup(){
        mkdir -p $SOURCEDIR
        cd $SOURCEDIR
        wget -O kernel.kly https://github.com/kendilinuxunuyap/kly-binary-packages/raw/master/kernel/kernel-6.10.8.kly
        tar -xf kernel.kly
        tar -xf rootfs.tar.xz
}
build(){
        echo ""
}
package(){
        cd $SOURCEDIR
        cp -prfv boot  ${DESTDIR}/
        cp -prfv lib/*  ${DESTDIR}/lib/
        find ${DESTDIR}/ -iname "*" -exec unxz {} \;
        ${DESTDIR}/sbin/ldconfig -r ${DESTDIR}          # sistem guncelleniyor
}
initsetup       # initsetup fonksiyonunu çalıştırır ve kaynak dosyayı indirir
setup           # setup fonksiyonu çalışır ve derleme öncesi kaynak dosyaların ayalanması sağlanır.
build           # build fonksiyonu çalışır ve kaynak dosyaları derlenir.
package         # package fonksiyonu çalışır, yükleme öncesi ayarlamalar yapılır ve yüklenir.
```

Kendi Kernelimiz Derleme

```
#-----------------------------------------------------------------------------------------------------------------------
#!/usr/bin/env bash
name="kernel-headers"
version="6.9.9"
description="Linux kernel"
source="https://cdn.kernel.org/pub/linux/kernel/v6.x/linux-$version.tar.xz"
depends="kernel"
builddepend="rsync,bc,cpio,gettext,elfutils,pahole,perl,python,tar,xz-utils"
display=":$(ls /tmp/.X11-unix/* | sed 's#/tmp/.X11-unix/X##' | head -1)"  # Display adı
user=$(who | grep "(${display})" | awk '{print $1}')                      # Display kullanıcısı

ROOT="/home/$user/distro"
ROOTBUILDDIR="$ROOT/build"                # Derleme dizini
BUILDDIR="$ROOT/build/build-${name}-${version}" # Alt dizin
DESTDIR="$ROOT/rootfs"                     # Yükleme dizini
PACKAGEDIR=$(pwd)                          # Paket dizini
SOURCEDIR="$ROOT/build/${name}-${version}" # Kaynak dizini

initsetup(){
        mkdir -p  $ROOTBUILDDIR #derleme dizini yoksa oluşturuluyor
        rm -rf $ROOTBUILDDIR/* #içeriği temizleniyor
        cd $ROOTBUILDDIR #dizinine geçiyoruz
        wget ${source}
        for f in *\ *; do mv "$f" "${f// /}"; done #isimde boşluk varsa silme işlemi yapılıyor
        dowloadfile=$(ls|head -1)
        filetype=$(file -b --extension $dowloadfile|cut -d'/' -f1)
        if [ "${filetype}" == "???" ]; then unzip  ${dowloadfile}; else tar -xvf ${dowloadfile};fi
        director=$(find ./* -maxdepth 0 -type d)
        directorname=$(basename ${director})
        if [ "${directorname}" != "${name}-${version}" ]; then mv $directorname ${name}-${version};fi
        mkdir -p $BUILDDIR&&mkdir -p $DESTDIR&&cd $SOURCEDIR
}
setup(){
        cp -prvf $PACKAGEDIR/files/ $SOURCEDIR/
        patch -Np1 -i $PACKAGEDIR/files/patch-$version
        cp $PACKAGEDIR/files/config $SOURCEDIR/.config
        make olddefconfig
}
build(){
        make bzImage
        make modules
}
```

```
#-------------------------------------------------------------------------------------------------------------------
# Kernel dereme scriptini devamı
package(){
        arch="x86"
        kernelbuilddir="${DESTDIR}/lib/modules/${version}/build"
        # install bzImage
        mkdir -p "$DESTDIR/boot"
        install -Dm644 "$(make -s image_name)" "$DESTDIR/boot/vmlinuz-${version}"
        #make INSTALL_PATH=$DESTDIR install ARCH=amd64
        # install modules
        mkdir -p ${DESTDIR}/lib/modules/${version}
        mkdir -p $DESTDIR/usr/src
        mkdir -p ${DESTDIR}/lib/modules/${version}/build
        make INSTALL_MOD_PATH=$DESTDIR modules_install INSTALL_MOD_STRIP=1 -j$(nproc)
        rm "${DESTDIR}/lib/modules/${version}"/{source,build} || true
        depmod --all --verbose --basedir="$DESTDIR" "${version}" || true
        # install build directories
        install .config "$DESTDIR/boot/config-${version}"
        install -Dt "$kernelbuilddir/kernel" -m644 kernel/Makefile
        install -Dt "$kernelbuilddir/arch/$arch" -m644 arch/$arch/Makefile
        cp -t "$kernelbuilddir" -a scripts
        install -Dt "$kernelbuilddir/tools/objtool" tools/objtool/objtool
        mkdir -p "$kernelbuilddir"/{fs/xfs,mm}
        ln -s "../../lib/modules/${version}/build" "$DESTDIR/usr/src/linux-headers-${version}"
        install -Dt "$kernelbuilddir" -m644 Makefile Module.symvers System.map vmlinux
        # install libc headers
        mkdir -p "$DESTDIR/usr/include/linux"
        cp -v -t "$DESTDIR/usr/include/" -a include/linux/
        cp -v -t "$DESTDIR/usr/" -a tools/include
        make headers_install INSTALL_HDR_PATH=$DESTDIR/usr

        mkdir -p "$kernelbuilddir" "$kernelbuilddir/arch/$arch"
        cp -v -t "$kernelbuilddir" -a include
        cp -v -t "$kernelbuilddir/arch/$arch" -a arch/$arch/include
        install -Dt "$kernelbuilddir/arch/$arch/kernel" -m644 arch/$arch/kernel/asm-offsets.*
        install -Dt "$kernelbuilddir/drivers/md" -m644 drivers/md/*.h
        install -Dt "$kernelbuilddir/net/mac80211" -m644 net/mac80211/*.h
        install -Dt "$kernelbuilddir/drivers/media/i2c" -m644 drivers/media/i2c/msp3400-driver.h
        install -Dt "$kernelbuilddir/drivers/media/usb/dvb-usb" -m644 drivers/media/usb/dvb-usb/*.h
        install -Dt "$kernelbuilddir/drivers/media/dvb-frontends" -m644 drivers/media/dvb-frontends/*.h
        install -Dt "$kernelbuilddir/drivers/media/tuners" -m644 drivers/media/tuners/*.h
        install -Dt "$kernelbuilddir/drivers/iio/common/hid-sensors" -m644 drivers/iio/common/hid-sensors/*.h
        # https://bugs.archlinux.org/task/71392
        find . -name 'Kconfig*' -exec install -Dm644 {} "$kernelbuilddir/{}" \;
        find -L "$kernelbuilddir" -type l -printf 'Removing %P\n' -delete
        # clearing
        find "$kernelbuilddir" -type f -name '*.o' -printf 'Removing %P\n' -delete
        if [[ -d "$kernelbuilddir" ]] ; then
    while read -rd '' file; do
        case "$(file -Sib "$file")" in
            application/x-sharedlib\;*)      # Libraries (.so)
                strip "$file" ;;
            application/x-executable\;*)     # Binaries
                strip "$file" ;;
            application/x-pie-executable\;*) # Relocatable binaries
                strip "$file" ;;
        esac
    done < <(find "$kernelbuilddir" -type f -perm -u+x ! -name vmlinux -print0)
        fi
        if [[ -f "$kernelbuilddir/vmlinux" ]] ; then
    strip "$kernelbuilddir/vmlinux"
        fi
        mkdir -p "$DESTDIR/usr/src"
        ln -sr "$kernelbuilddir" "$DESTDIR/usr/src/linux"
    mv -vf System.map $DESTDIR/boot/System.map-$version
    find ${DESTDIR}/ -iname "*" -exec unxz {} \;
    depmod -b "$DESTDIR" -F $DESTDIR/boot/System.map-$version $version
}
initsetup       # initsetup fonksiyonunu çalıştırır ve kaynak dosyayı indirir
setup           # setup fonksiyonu çalışır ve derleme öncesi kaynak dosyaların ayalanması sağlanır.
build           # build fonksiyonu çalışır ve kaynak dosyaları derlenir.
package         # package fonksiyonu çalışır, yükleme öncesi ayarlamalar yapılır ve yüklenir.
```

Ek dosyaları indirmek için tıklayınız. **Paket Derleme Yöntemi** konusunda anlatıldığı gibi derleme işlemini yapınız.

dialog

Dialog paketi, terminalde metin kutuları, onay kutuları, seçim kutuları gibi farklı türde diyaloglar oluşturmamızı sağlayan kütüphanedir.

Derleme

```
#-----------------------------------------------------------------------------------------------------------------------
#!/usr/bin/env bash
version="1.3-20230209"
name="dialog"
depends="glibc,readline,ncurses"
description="shell box kütüphanesi"
source="https://invisible-island.net/archives/dialog/${name}-${version}.tgz"
display=":$(ls /tmp/.X11-unix/* | sed 's#/tmp/.X11-unix/X##' | head -1)"  # Display adı
user=$(who | grep "(${display})" | awk '{print $1}')                      # Display kullanıcısı

ROOT="/home/$user/distro"
ROOTBUILDDIR="$ROOT/build"                # Derleme dizini
BUILDDIR="$ROOT/build/build-${name}-${version}" # Alt dizin
DESTDIR="$ROOT/rootfs"                     # Yükleme dizini
PACKAGEDIR=$(pwd)                          # Paket dizini
SOURCEDIR="$ROOT/build/${name}-${version}" # Kaynak dizini

initsetup(){
            mkdir -p  $ROOTBUILDDIR #derleme dizini yoksa oluşturuluyor
            rm -rf $ROOTBUILDDIR/* #içeriği temizleniyor
            cd $ROOTBUILDDIR #dizinine geçiyoruz
    wget ${source}
    for f in *\ *; do mv "$f" "${f// /}"; done #isimde boşluk varsa silme işlemi yapılıyor
            dowloadfile=$(ls|head -1)
            filetype=$(file -b --extension $dowloadfile|cut -d'/' -f1)
            if [ "${filetype}" == "???" ]; then unzip  ${dowloadfile}; else tar -xvf ${dowloadfile};fi
            director=$(find ./* -maxdepth 0 -type d)
            directorname=$(basename ${director})
            if [ "${directorname}" != "${name}-${version}" ]; then mv $directorname ${name}-${version};fi
            mkdir -p $BUILDDIR&&mkdir -p $DESTDIR&&cd $SOURCEDIR
}

setup(){
./configure --prefix=/usr --libdir=/lib64/ --with-ncursesw
}
build(){
        make
}
package(){
        make install DESTDIR=$DESTDIR
        ${DESTDIR}/sbin/ldconfig -r ${DESTDIR}           # sistem guncelleniyor
}
initsetup       # initsetup fonksiyonunu çalıştırır ve kaynak dosyayı indirir
setup           # setup fonksiyonu çalışır ve derleme öncesi kaynak dosyaların ayalanması sağlanır.
build           # build fonksiyonu çalışır ve kaynak dosyaları derlenir.
package         # package fonksiyonu çalışır, yükleme öncesi ayarlamalar yapılır ve yüklenir.
```

Paket Derleme Yöntemi konusunda anlatıldığı gibi derleme işlemini yapınız.

live-boot

Live-boot paketi, işletim sistemini kurmadan çalıştıran bir araçtır.

Debian ortamında bu paketin derlenmesi için; **sudo apt install po4a** komutuyla paketin kurulması gerekmektedir.

```
#-----------------------------------------------------------------------------------------------------------------------
#!/usr/bin/env bash
version="1230131"
name="live-boot"
depends="glibc,acl,openssl"
description="shell ve network copy"
source="https://salsa.debian.org/live-team/${name}/-/archive/debian/1%2520230131/${name}-debian-1%2520230131.tar.gz"
display=":$(ls /tmp/.X11-unix/* | sed 's#/tmp/.X11-unix/X##' | head -1)"  # Display adı
user=$(who | grep "(${display})" | awk '{print $1}')                      # Display kullanıcısı

ROOT="/home/$user/distro"
ROOTBUILDDIR="$ROOT/build"                # Derleme dizini
BUILDDIR="$ROOT/build/build-${name}-${version}" # Alt dizin
DESTDIR="$ROOT/rootfs"                     # Yükleme dizini
PACKAGEDIR=$(pwd)                          # Paket dizini
SOURCEDIR="$ROOT/build/${name}-${version}" # Kaynak dizini

initsetup(){
            mkdir -p  $ROOTBUILDDIR #derleme dizini yoksa oluşturuluyor
            rm -rf $ROOTBUILDDIR/* #içeriği temizleniyor
            cd $ROOTBUILDDIR #dizinine geçiyoruz
    wget ${source}
    for f in *\ *; do mv "$f" "${f// /}"; done #isimde boşluk varsa silme işlemi yapılıyor
            dowloadfile=$(ls|head -1)
            filetype=$(file -b --extension $dowloadfile|cut -d'/' -f1)
            if [ "${filetype}" == "???" ]; then unzip  ${dowloadfile}; else tar -xvf ${dowloadfile};fi
            director=$(find ./* -maxdepth 0 -type d)
            directorname=$(basename ${director})
            if [ "${directorname}" != "${name}-${version}" ]; then mv $directorname ${name}-${version};fi
            mkdir -p $BUILDDIR&&mkdir -p $DESTDIR&&cd $SOURCEDIR
}
setup(){
        echo ""
}
build(){
        make
}
package(){
        make install DESTDIR=$DESTDIR
        sed -i "s/copy_exec \/bin\/mount \/bin/copy_exec \/usr\/bin\/mount \/bin/g" \
        $DESTDIR/usr/share/initramfs-tools/hooks/live
        sed -i 's|cp -a "\${FILE}" "\${DESTDIR}/\${FILE}"|cp -r --no-preserve=ownership "\${FILE}" \
        "\${DESTDIR}/\${FILE}"|g' $DESTDIR/usr/share/initramfs-tools/hooks/live
        ${DESTDIR}/sbin/ldconfig -r ${DESTDIR}          # sistem guncelleniyor
}
initsetup       # initsetup fonksiyonunu çalıştırır ve kaynak dosyayı indirir
setup           # setup fonksiyonu çalışır ve derleme öncesi kaynak dosyaların ayalanması sağlanır.
build           # build fonksiyonu çalışır ve kaynak dosyaları derlenir.
package         # package fonksiyonu çalışır, yükleme öncesi ayarlamalar yapılır ve yüklenir.
```

Paket Derleme Yöntemi konusunda anlatıldığı gibi derleme işlemini yapınız.

live-config

Live-Config, canlı sistem açılırken kullanılan ayarları yapılandırma paketidir.

Derleme

```
#-----------------------------------------------------------------------------------------------------------------------
#!/usr/bin/env bash
version="11.0.4"
name="live-config"
depends="glibc,acl,openssl"
description="shell ve network copy"
source="https://salsa.debian.org/live-team/live-config/-/archive/debian/11.0.4/live-config-debian-11.0.4.tar.gz"
display=":$(ls /tmp/.X11-unix/* | sed 's#/tmp/.X11-unix/X##' | head -1)"  # Display adı
user=$(who | grep "(${display})" | awk '{print $1}')                      # Display kullanıcısı

ROOT="/home/$user/distro"
ROOTBUILDDIR="$ROOT/build"                # Derleme dizini
BUILDDIR="$ROOT/build/build-${name}-${version}" # Alt dizin
DESTDIR="$ROOT/rootfs"                     # Yükleme dizini
PACKAGEDIR=$(pwd)                          # Paket dizini
SOURCEDIR="$ROOT/build/${name}-${version}" # Kaynak dizini

initsetup(){
            mkdir -p  $ROOTBUILDDIR #derleme dizini yoksa oluşturuluyor
            rm -rf $ROOTBUILDDIR/* #içeriği temizleniyor
            cd $ROOTBUILDDIR #dizinine geçiyoruz
                wget ${source}
                for f in *\ *; do mv "$f" "${f// /}"; done #isimde boşluk varsa silme işlemi yapılıyor
            dowloadfile=$(ls|head -1)
            filetype=$(file -b --extension $dowloadfile|cut -d'/' -f1)
            if [ "${filetype}" == "???" ]; then unzip  ${dowloadfile}; else tar -xvf ${dowloadfile};fi
            director=$(find ./* -maxdepth 0 -type d)
            directorname=$(basename ${director})
            if [ "${directorname}" != "${name}-${version}" ]; then mv $directorname ${name}-${version};fi
            mkdir -p $BUILDDIR&&mkdir -p $DESTDIR&&cd $SOURCEDIR
}
setup(){
        echo ""
}
build(){
        make
}
package(){
        make install DESTDIR=$DESTDIR
        ${DESTDIR}/sbin/ldconfig -r ${DESTDIR}           # sistem guncelleniyor
}
initsetup       # initsetup fonksiyonunu çalıştırır ve kaynak dosyayı indirir
setup           # setup fonksiyonu çalışır ve derleme öncesi kaynak dosyaların ayalanması sağlanır.
build           # build fonksiyonu çalışır ve kaynak dosyaları derlenir.
package         # package fonksiyonu çalışır, yükleme öncesi ayarlamalar yapılır ve yüklenir.
```

Paket Derleme Yöntemi konusunda anlatıldığı gibi derleme işlemini yapınız.

parted

Parted, disk bölümlerini oluşturmak, silmek, boyutlandırmak ve düzenlemek için kullanılan bir komut satırı aracıdır.

Derleme

Debian ortamında bu paketin derlenmesi için;

- **sudo apt install libparted-dev**

komutuyla paketin kurulması gerekmektedir.

```
#-----------------------------------------------------------------------------------------------------------------------
#!/usr/bin/env bash
version="3.6"
name="parted"
depends="glibc"
description="disks tools"
source="https://ftp.gnu.org/gnu/parted/parted-${version}.tar.xz"
display=":$(ls /tmp/.X11-unix/* | sed 's#/tmp/.X11-unix/X##' | head -1)"  # Display adı
user=$(who | grep "(${display})" | awk '{print $1}')                      # Display kullanıcısı

ROOT="/home/$user/distro"
ROOTBUILDDIR="$ROOT/build"                # Derleme dizini
BUILDDIR="$ROOT/build/build-${name}-${version}" # Alt dizin
DESTDIR="$ROOT/rootfs"                     # Yükleme dizini
PACKAGEDIR=$(pwd)                          # Paket dizini
SOURCEDIR="$ROOT/build/${name}-${version}" # Kaynak dizini

initsetup(){
            mkdir -p  $ROOTBUILDDIR #derleme dizini yoksa oluşturuluyor
            rm -rf $ROOTBUILDDIR/* #içeriği temizleniyor
            cd $ROOTBUILDDIR #dizinine geçiyoruz
            wget ${source}
            for f in *\ *; do mv "$f" "${f// /}"; done #isimde boşluk varsa silme işlemi yapılıyor
            dowloadfile=$(ls|head -1)
            filetype=$(file -b --extension $dowloadfile|cut -d'/' -f1)
            if [ "${filetype}" == "???" ]; then unzip  ${dowloadfile}; else tar -xvf ${dowloadfile};fi
            director=$(find ./* -maxdepth 0 -type d)
            directorname=$(basename ${director})
            if [ "${directorname}" != "${name}-${version}" ]; then mv $directorname ${name}-${version};fi
            mkdir -p $BUILDDIR&&mkdir -p $DESTDIR&&cd $SOURCEDIR
}

setup(){
./configure --prefix=/usr --libdir=/usr/lib64/ --sbindir=/usr/bin --disable-rpath --disable-device-mapper
}
build(){
        make
}
package(){
        make install DESTDIR=$DESTDIR
        ${DESTDIR}/sbin/ldconfig -r ${DESTDIR}           # sistem guncelleniyor
}
initsetup       # initsetup fonksiyonunu çalıştırır ve kaynak dosyayı indirir
setup           # setup fonksiyonu çalışır ve derleme öncesi kaynak dosyaların ayalanması sağlanır.
build           # build fonksiyonu çalışır ve kaynak dosyaları derlenir.
package         # package fonksiyonu çalışır, yükleme öncesi ayarlamalar yapılır ve yüklenir.
```

Paket Derleme Yöntemi konusunda anlatıldığı gibi derleme işlemini yapınız.

busybox

BusyBox, çeşitli komutları tek bir ikili dosya altında toplar. BusyBox, ls, cp, mv, rm gibi temel komutları, ağ yönetimi, dosya sistemleri ve sistem yönetimi gerekli araçları içinde barındıran tek ikili dosyadır.

Örneğin; Kullanıcı, BusyBox ile bir dosyayı kopyalamak için aşağıdaki komut kullanılabilir:

```
busybox cp kaynak_dosya hedef_dosya
```

Derleme

```
#-----------------------------------------------------------------------------------------------------------------------
#!/usr/bin/env bash
version="1.36.1"
name="busybox"
depends="glibc"
description="linux araç paketi static derlenmiş hali"
source="https://busybox.net/downloads/${name}-${version}.tar.bz2"
display=":$(ls /tmp/.X11-unix/* | sed 's#/tmp/.X11-unix/X##' | head -1)"  # Display adı
user=$(who | grep "(${display})" | awk '{print $1}')                      # Display kullanıcısı

ROOT="/home/$user/distro"
ROOTBUILDDIR="$ROOT/build"                # Derleme dizini
BUILDDIR="$ROOT/build/build-${name}-${version}" # Alt dizin
DESTDIR="$ROOT/rootfs"                     # Yükleme dizini
PACKAGEDIR=$(pwd)                          # Paket dizini
SOURCEDIR="$ROOT/build/${name}-${version}" # Kaynak dizini

initsetup(){
        mkdir -p  $ROOTBUILDDIR #derleme dizini yoksa oluşturuluyor
        rm -rf $ROOTBUILDDIR/* #içeriği temizleniyor
        cd $ROOTBUILDDIR #dizinine geçiyoruz
        wget ${source}
        for f in *\ *; do mv "$f" "${f// /}"; done #isimde boşluk varsa silme işlemi yapılıyor
        dowloadfile=$(ls|head -1)
        filetype=$(file -b --extension $dowloadfile|cut -d'/' -f1)
        if [ "${filetype}" == "???" ]; then unzip  ${dowloadfile}; else tar -xvf ${dowloadfile};fi
        director=$(find ./* -maxdepth 0 -type d)
        directorname=$(basename ${director})
        if [ "${directorname}" != "${name}-${version}" ]; then mv $directorname ${name}-${version};fi
        mkdir -p $BUILDDIR&&mkdir -p $DESTDIR&&cd $SOURCEDIR
}
setup(){
        cp -prfv $PACKAGEDIR/files $SOURCEDIR/
        make defconfig
        sed -i "s|.*CONFIG_STATIC_LIBGCC .*|CONFIG_STATIC_LIBGCC=y|" .config
        sed -i "s|.*CONFIG_STATIC .*|CONFIG_STATIC=y|" .config
}
build(){
        make
}
package(){
        mkdir -p $DESTDIR/bin
        install busybox ${DESTDIR}/bin/busybox
        # install udhcpc script and service
        mkdir -p ${DESTDIR}/usr/share/udhcpc/ ${DESTDIR}/etc/init.d/
        install ../files/udhcpc.script ${DESTDIR}/usr/share/udhcpc/default.script
        install ../files/udhcpc.openrc ${DESTDIR}/etc/init.d/udhcpc
        cd $DESTDIR/bin&&ln -s busybox hostname
        cd $DESTDIR/bin&&ln -s busybox udhcpc
        cd $DESTDIR/bin&&ln -s busybox ps
        cd $DESTDIR/bin&&ln -s busybox which
        ${DESTDIR}/sbin/ldconfig -r ${DESTDIR}           # sistem guncelleniyor
}
initsetup       # initsetup fonksiyonunu çalıştırır ve kaynak dosyayı indirir
setup           # setup fonksiyonu çalışır ve derleme öncesi kaynak dosyaların ayalanması sağlanır.
build           # build fonksiyonu çalışır ve kaynak dosyaları derlenir.
package         # package fonksiyonu çalışır, yükleme öncesi ayarlamalar yapılır ve yüklenir.
```

Ek dosyaları indirmek için tıklayınız. **Paket Derleme Yöntemi** konusunda anlatıldığı gibi derleme işlemini yapınız.

nano

Nano, terminal tabanlı bir metin düzenleme editörüdür.

Derleme

```
#-----------------------------------------------------------------------------------------------------------------------
#!/usr/bin/env bash
version="7.2"
name="nano"
depends="glibc,readline,ncurses,file"
description="şıkıştırma kütüphanesi"
source="https://www.nano-editor.org/dist/v7/${name}-${version}.tar.xz"
display=":$(ls /tmp/.X11-unix/* | sed 's#/tmp/.X11-unix/X##' | head -1)"  # Display adı
user=$(who | grep "(${display})" | awk '{print $1}')                      # Display kullanıcısı

ROOT="/home/$user/distro"
ROOTBUILDDIR="$ROOT/build"                # Derleme dizini
BUILDDIR="$ROOT/build/build-${name}-${version}" # Alt dizin
DESTDIR="$ROOT/rootfs"                     # Yükleme dizini
PACKAGEDIR=$(pwd)                          # Paket dizini
SOURCEDIR="$ROOT/build/${name}-${version}" # Kaynak dizini

initsetup(){
            mkdir -p  $ROOTBUILDDIR #derleme dizini yoksa oluşturuluyor
            rm -rf $ROOTBUILDDIR/* #içeriği temizleniyor
            cd $ROOTBUILDDIR #dizinine geçiyoruz
            wget ${source}
            for f in *\ *; do mv "$f" "${f// /}"; done #isimde boşluk varsa silme işlemi yapılıyor
            dowloadfile=$(ls|head -1)
            filetype=$(file -b --extension $dowloadfile|cut -d'/' -f1)
            if [ "${filetype}" == "???" ]; then unzip  ${dowloadfile}; else tar -xvf ${dowloadfile};fi
            director=$(find ./* -maxdepth 0 -type d)
            directorname=$(basename ${director})
            if [ "${directorname}" != "${name}-${version}" ]; then mv $directorname ${name}-${version};fi
            mkdir -p $BUILDDIR&&mkdir -p $DESTDIR&&cd $SOURCEDIR
}

setup(){
        ./configure --prefix=/usr
}
build(){
        make
}
package(){
        make install DESTDIR=$DESTDIR
        cd $DESTDIR
        mkdir -p $DESTDIR/lib
        echo "INPUT(-lncursesw)" > $DESTDIR/lib/libncurses.so
        ${DESTDIR}/sbin/ldconfig -r ${DESTDIR}           # sistem guncelleniyor
}
initsetup       # initsetup fonksiyonunu çalıştırır ve kaynak dosyayı indirir
setup           # setup fonksiyonu çalışır ve derleme öncesi kaynak dosyaların ayalanması sağlanır.
build           # build fonksiyonu çalışır ve kaynak dosyaları derlenir.
package         # package fonksiyonu çalışır, yükleme öncesi ayarlamalar yapılır ve yüklenir.
```

Paket Derleme Yöntemi konusunda anlatıldığı gibi derleme işlemini yapınız.

grub

GRUB, bilgisayar açılırken işletim sistemini başlatan bir önyükleyici (boot loader) programıdır. GRUB yapılandırma dosyası genellikle /boot/grub/grub.cfg konumunda bulunur.

Derleme

```
#-----------------------------------------------------------------------------------------------------------------------
#!/usr/bin/env bash
name="grub"
version="2.12"
description="GNU GRand Unified Bootloader"
source="https://ftp.gnu.org/gnu/grub/grub-$version.tar.xz"
depends="glibc,readline,ncurses,xz-utils,efibootmgr"
builddepend="rsync,freetype,ttf-dejavu"
group="sys.boot"
uses=(efi bios)
uses_extra=(ia32)
dontstrip=1
efi_dp=(efibootmgr)
ia32_dp=(efibootmgr)
unset CFLAGS
unset CXXFLAGS
get_grub_opt(){ echo -n "--disable-efiemu "
        if [[ "$1" == "efi" ]] ; then echo -n "--with-platform=efi --target=x86_64"
        elif [[ "$1" == "ia32" ]] ; then echo -n "--with-platform=efi --target=i386"
        elif [[ "$1" == "bios" ]] ; then echo -n "--with-platform=pc"
        fi
        }
display=":$(ls /tmp/.X11-unix/* | sed 's#/tmp/.X11-unix/X##' | head -1)"  # Display adı
user=$(who | grep "(${display})" | awk '{print $1}')                      # Display kullanıcısı
ROOT="/home/$user/distro"
ROOTBUILDDIR="$ROOT/build"                # Derleme dizini
BUILDDIR="$ROOT/build/build-${name}-${version}" # Alt dizin
DESTDIR="$ROOT/rootfs"                     # Yükleme dizini
PACKAGEDIR=$(pwd)                          # Paket dizini
SOURCEDIR="$ROOT/build/${name}-${version}" # Kaynak dizini

initsetup(){
                mkdir -p  $ROOTBUILDDIR #derleme dizini yoksa oluşturuluyor
                rm -rf $ROOTBUILDDIR/* #içeriği temizleniyor
                cd $ROOTBUILDDIR #dizinine geçiyoruz
                wget ${source}
                for f in *\ *; do mv "$f" "${f// /}"; done #isimde boşluk varsa silme işlemi yapılıyor
                dowloadfile=$(ls|head -1)
                filetype=$(file -b --extension $dowloadfile|cut -d'/' -f1)
                if [ "${filetype}" == "???" ]; then unzip  ${dowloadfile}; else tar -xvf ${dowloadfile};fi
                director=$(find ./* -maxdepth 0 -type d)
                directorname=$(basename ${director})
                if [ "${directorname}" != "${name}-${version}" ]; then mv $directorname ${name}-${version};fi
                mkdir -p $BUILDDIR&&mkdir -p $DESTDIR&&cd $SOURCEDIR
}
setup(){        cd $ROOTBUILDDIR
 echo depends bli part_gpt > $SOURCEDIR/grub-core/extra_deps.lst
        for tgt in ${uses[@]} ; do cp -prfv $name-$version $tgt; done
        for tgt in ${uses[@]} ; do
            cd $tgt
            autoreconf -fvi
            ./configure --prefix=/usr --sysconfdir=/etc --libdir=/usr/lib64/ --disable-nls --disable-werror \
            --disable-grub-themes $(get_grub_opt $tgt)
            cd ..
        done
}
build(){ for tgt in ${uses[@]} ; do make -C $tgt; done  }
package(){
        for tgt in ${uses[@]} ; do make $jobs -C $tgt install DESTDIR=$DESTDIR; done
        mkdir -p $DESTDIR/etc/default $DESTDIR/usr/bin/
        install $PACKAGEDIR/files/grub $DESTDIR/etc/default/grub
        install -vDm 755  $PACKAGEDIR/files/update-grub $DESTDIR/usr/bin/update-grub
        ${DESTDIR}/sbin/ldconfig -r ${DESTDIR}           # sistem guncelleniyor
}
initsetup       # initsetup fonksiyonunu çalıştırır ve kaynak dosyayı indirir
setup           # setup fonksiyonu çalışır ve derleme öncesi kaynak dosyaların ayalanması sağlanır.
build           # build fonksiyonu çalışır ve kaynak dosyaları derlenir.
package         # package fonksiyonu çalışır, yükleme öncesi ayarlamalar yapılır ve yüklenir.
```

Ek dosyaları indirmek için tıklayınız. **Paket Derleme Yöntemi** konusunda anlatıldığı gibi derleme işlemini yapınız.

efibootmgr

efibootmgr, UEFI sistemlerde önyükleme aracıdır.

Derleme

Debian ortamında bu paketin derlenmesi için;

- **sudo apt install libefiboot-dev libefivar-dev libpopt-dev** komutuyla paketin kurulması gerekmektedir.

```
#-------------------------------------------------------------------------------------------------------------------
#!/usr/bin/env bash
name="efibootmgr"
version="16"
description="Linux user-space application to modify the Intel Extensible Firmware Interface (EFI) Boot Manager."
source="https://github.com/rhboot/efibootmgr/archive/refs/tags/$version.tar.gz"
depends="efivar,popt"
builddepend=""
display=":$(ls /tmp/.X11-unix/* | sed 's#/tmp/.X11-unix/X##' | head -1)"  # Display adı
user=$(who | grep "(${display})" | awk '{print $1}')                      # Display kullanıcısı

ROOT="/home/$user/distro"
ROOTBUILDDIR="$ROOT/build"                # Derleme dizini
BUILDDIR="$ROOT/build/build-${name}-${version}" # Alt dizin
DESTDIR="$ROOT/rootfs"                     # Yükleme dizini
PACKAGEDIR=$(pwd)                          # Paket dizini
SOURCEDIR="$ROOT/build/${name}-${version}" # Kaynak dizini

initsetup(){
            mkdir -p  $ROOTBUILDDIR #derleme dizini yoksa oluşturuluyor
            rm -rf $ROOTBUILDDIR/* #içeriği temizleniyor
            cd $ROOTBUILDDIR #dizinine geçiyoruz
    wget ${source}
    for f in *\ *; do mv "$f" "${f// /}"; done #isimde boşluk varsa silme işlemi yapılıyor
            dowloadfile=$(ls|head -1)
            filetype=$(file -b --extension $dowloadfile|cut -d'/' -f1)
            if [ "${filetype}" == "???" ]; then unzip  ${dowloadfile}; else tar -xvf ${dowloadfile};fi
            director=$(find ./* -maxdepth 0 -type d)
            directorname=$(basename ${director})
            if [ "${directorname}" != "${name}-${version}" ]; then mv $directorname ${name}-${version};fi
            mkdir -p $BUILDDIR&&mkdir -p $DESTDIR&&cd $SOURCEDIR
}
setup(){
        echo ""
}
build(){
    make sbindir=/usr/bin EFIDIR=/boot/efi PCDIR=/usr/lib64/pkgconfig
}
package(){
        EFIDIR="/boot/efi" sbindir=/usr/bin make DESTDIR="$DESTDIR" install
        ${DESTDIR}/sbin/ldconfig -r ${DESTDIR}           # sistem guncelleniyor
}

initsetup       # initsetup fonksiyonunu çalıştırır ve kaynak dosyayı indirir
setup           # setup fonksiyonu çalışır ve derleme öncesi kaynak dosyaların ayalanması sağlanır.
build           # build fonksiyonu çalışır ve kaynak dosyaları derlenir.
package         # package fonksiyonu çalışır, yükleme öncesi ayarlamalar yapılır ve yüklenir.
```

Paket Derleme Yöntemi konusunda anlatıldığı gibi derleme işlemini yapınız.

efivar

efivar paketi, UEFI sistemlerde, UEFI değişkenlerini okuma, yazma ve silme işlemlerini gerçekleştiriren kütüphanedir.

Derleme

Debian ortamında bu paketin derlenmesi için; **sudo apt install libefivar-dev** komutuyla paketin kurulması gerekmektedir.

```
#-----------------------------------------------------------------------------------------------------------------------
#!/usr/bin/env bash
name="efivar"
version="39"
description="Tools and libraries to work with EFI variables"
source="https://github.com/rhboot/efivar/archive/refs/tags/$version.tar.gz"
depends=""
builddepend=""
display=":$(ls /tmp/.X11-unix/* | sed 's#/tmp/.X11-unix/X##' | head -1)"  # Display adı
user=$(who | grep "(${display})" | awk '{print $1}')                      # Display kullanıcısı

ROOT="/home/$user/distro"
ROOTBUILDDIR="$ROOT/build"                # Derleme dizini
BUILDDIR="$ROOT/build/build-${name}-${version}" # Alt dizin
DESTDIR="$ROOT/rootfs"                     # Yükleme dizini
PACKAGEDIR=$(pwd)                          # Paket dizini
SOURCEDIR="$ROOT/build/${name}-${version}" # Kaynak dizini

initsetup(){
            mkdir -p  $ROOTBUILDDIR #derleme dizini yoksa oluşturuluyor
            rm -rf $ROOTBUILDDIR/* #içeriği temizleniyor
            cd $ROOTBUILDDIR #dizinine geçiyoruz
    wget ${source}
    for f in *\ *; do mv "$f" "${f// /}"; done #isimde boşluk varsa silme işlemi yapılıyor
            dowloadfile=$(ls|head -1)
            filetype=$(file -b --extension $dowloadfile|cut -d'/' -f1)
            if [ "${filetype}" == "???" ]; then unzip  ${dowloadfile}; else tar -xvf ${dowloadfile};fi
            director=$(find ./* -maxdepth 0 -type d)
            directorname=$(basename ${director})
            if [ "${directorname}" != "${name}-${version}" ]; then mv $directorname ${name}-${version};fi
            mkdir -p $BUILDDIR&&mkdir -p $DESTDIR&&cd $SOURCEDIR
}
setup(){
        export ERRORS=''
        export PATH=$PATH:$HOME
    echo "exit 0" > $HOME/mandoc               # fake mandoc for ignore extra dependency
    chmod +x $HOME/mandoc
}
build(){
    make
}
package(){
        local make_options=(V=1 libdir=/usr/lib64/ bindir=/usr/bin/ mandir=/usr/share/man/ includedir=/usr/include/)
    make DESTDIR=$DESTDIR "${make_options[@]}" install
    ${DESTDIR}/sbin/ldconfig -r ${DESTDIR}           # sistem guncelleniyor
}
initsetup       # initsetup fonksiyonunu çalıştırır ve kaynak dosyayı indirir
setup           # setup fonksiyonu çalışır ve derleme öncesi kaynak dosyaların ayalanması sağlanır.
build           # build fonksiyonu çalışır ve kaynak dosyaları derlenir.
package         # package fonksiyonu çalışır, yükleme öncesi ayarlamalar yapılır ve yüklenir.
```

Paket Derleme Yöntemi konusunda anlatıldığı gibi derleme işlemini yapınız.

libssh

libssh, programların SSH (Secure Shell) protokolü uzak sistemlere şifreli veri iletimi ve komut çalıştırma işlemleri yapmamızı sağlayan kütüphanedir.

Derleme

```
#-----------------------------------------------------------------------------------------------------------------------
#!/usr/bin/env bash
name="libssh"
version="0.10.4"
description="C library implenting the SSHv2 protocol on client and server side"
source=("https://www.libssh.org/files/0.10/libssh-${version}.tar.xz")
depends="openssl,zlib"
display=":$(ls /tmp/.X11-unix/* | sed 's#/tmp/.X11-unix/X##' | head -1)"  # Display adı
user=$(who | grep "(${display})" | awk '{print $1}')                      # Display kullanıcısı

ROOT="/home/$user/distro"
ROOTBUILDDIR="$ROOT/build"                # Derleme dizini
BUILDDIR="$ROOT/build/build-${name}-${version}" # Alt dizin
DESTDIR="$ROOT/rootfs"                     # Yükleme dizini
PACKAGEDIR=$(pwd)                          # Paket dizini
SOURCEDIR="$ROOT/build/${name}-${version}" # Kaynak dizini

initsetup(){
            mkdir -p  $ROOTBUILDDIR #derleme dizini yoksa oluşturuluyor
            rm -rf $ROOTBUILDDIR/* #içeriği temizleniyor
            cd $ROOTBUILDDIR #dizinine geçiyoruz
    wget ${source}
    for f in *\ *; do mv "$f" "${f// /}"; done #isimde boşluk varsa silme işlemi yapılıyor
            dowloadfile=$(ls|head -1)
            filetype=$(file -b --extension $dowloadfile|cut -d'/' -f1)
            if [ "${filetype}" == "???" ]; then unzip  ${dowloadfile}; else tar -xvf ${dowloadfile};fi
            director=$(find ./* -maxdepth 0 -type d)
            directorname=$(basename ${director})
            if [ "${directorname}" != "${name}-${version}" ]; then mv $directorname ${name}-${version};fi
            mkdir -p $BUILDDIR&&mkdir -p $DESTDIR&&cd $SOURCEDIR
}

setup() {
    cmake -S $SOURCEDIR -B $BUILDDIR -DCMAKE_INSTALL_PREFIX=/usr -DCMAKE_INSTALL_LIBDIR=/usr/lib64 \
        -DWITH_EXAMPLES=NO -DBUILD_SHARED_LIBS=YES -DBUILD_STATIC_LIB=YES -DWITH_NACL=OFF \
        -DWITH_GCRYPT=OFF -DWITH_MBEDTLS=OFF -DWITH_GSSAPI=OFF \
        -DWITH_PCAP=OFF -DWITH_SERVER=ON -DWITH_SFTP=ON -DWITH_ZLIB=ON
}
build() {
    make -C $BUILDDIR
}
package() {
    make -C $BUILDDIR install DESTDIR=$DESTDIR
    install $BUILDDIR/src/libssh.a ${DESTDIR}/usr/lib64/
    ${DESTDIR}/sbin/ldconfig -r ${DESTDIR}           # sistem guncelleniyor
}
initsetup       # initsetup fonksiyonunu çalıştırır ve kaynak dosyayı indirir
setup           # setup fonksiyonu çalışır ve derleme öncesi kaynak dosyaların ayalanması sağlanır.
build           # build fonksiyonu çalışır ve kaynak dosyaları derlenir.
package         # package fonksiyonu çalışır, yükleme öncesi ayarlamalar yapılır ve yüklenir.
```

Paket Derleme Yöntemi konusunda anlatıldığı gibi derleme işlemini yapınız.

openssh

OpenSSH, uzak sunuculara bağlanma, dosya taransferi ve uzaktan komut çalıştırmalarını sağlar. OpenSSH, ssh, scp, sftp ve sshd gibi araçları içerir.

Debian ortamında bu paketin derlenmesi için; **sudo apt install libssh2-1-dev libcrypt-dev** komutuyla paketlerin kurulması gerekmektedir.

```
#-----------------------------------------------------------------------------------------------------------------------
#!/usr/bin/env bash
name="openssh"
version="9.6p1"
description="OpenBSD ssh server & client"
source="https://ftp.openbsd.org/pub/OpenBSD/OpenSSH/portable/openssh-$version.tar.gz"
depends="zlib,libxcrypt,openssl,libmd,libssh"
display=":$(ls /tmp/.X11-unix/* | sed 's#/tmp/.X11-unix/X##' | head -1)"  # Display adı
user=$(who | grep "(${display})" | awk '{print $1}')                      # Display kullanıcısı

ROOT="/home/$user/distro"
ROOTBUILDDIR="$ROOT/build"                # Derleme dizini
BUILDDIR="$ROOT/build/build-${name}-${version}" # Alt dizin
DESTDIR="$ROOT/rootfs"                     # Yükleme dizini
PACKAGEDIR=$(pwd)                          # Paket dizini
SOURCEDIR="$ROOT/build/${name}-${version}" # Kaynak dizini

initsetup(){
        mkdir -p  $ROOTBUILDDIR #derleme dizini yoksa oluşturuluyor
        rm -rf $ROOTBUILDDIR/* #içeriği temizleniyor
        cd $ROOTBUILDDIR #dizinine geçiyoruz
        wget ${source}
        for f in *\ *; do mv "$f" "${f// /}"; done #isimde boşluk varsa silme işlemi yapılıyor
        dowloadfile=$(ls|head -1)
        filetype=$(file -b --extension $dowloadfile|cut -d'/' -f1)
        if [ "${filetype}" == "???" ]; then unzip  ${dowloadfile}; else tar -xvf ${dowloadfile};fi
        director=$(find ./* -maxdepth 0 -type d)
        directorname=$(basename ${director})
        if [ "${directorname}" != "${name}-${version}" ]; then mv $directorname ${name}-${version};fi
        mkdir -p $BUILDDIR&&mkdir -p $DESTDIR&&cd $SOURCEDIR
}
setup(){
        cp -prfv $PACKAGEDIR/files $SOURCEDIR/
        ./configure --prefix=/usr --libdir=/usr/lib64/ --sysconfdir=/etc/ssh --without-pam --disable-strip \
            --with-ssl-engine --with-privsep-user=nobody --with-pid-dir=/run  \
            --with-default-path="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
}
build(){
        make
}
package(){
        make install DESTDIR=$DESTDIR
        mkdir -p "$DESTDIR"/etc/{passwd,group,init,conf}.d
        install -m755 -D $SOURCEDIR/files/sshd.initd "$DESTDIR"/etc/init.d/sshd
        install $SOURCEDIR/files/sshd.initd  ${DESTDIR}/etc/runlevels/default/sshd
        install -m755 -D $SOURCEDIR/files/sshd.confd "$DESTDIR"/etc/conf.d/sshd
        ${DESTDIR}/sbin/ldconfig -r ${DESTDIR}           # sistem guncelleniyor
        sed -i "/nobody/d"  ${DESTDIR}/etc/group
        sed -i "/nobody/d"  ${DESTDIR}/etc/passwd
        mkdir -p  ${DESTDIR}/var/empty
        chown root:root  ${DESTDIR}/var/empty
        chmod 755  ${DESTDIR}/var/empty
        echo "nobody:!:65534:" >>  ${DESTDIR}/etc/group
        echo "nobody:!:65534:65534::/var/empty:/usr/sbin/nologin" >>  ${DESTDIR}/etc/passwd
        sed -i "/PermitRootLogin/d" ${DESTDIR}/etc/ssh/sshd_config
        echo -e "\nPermitRootLogin yes">> ${DESTDIR}/etc/ssh/sshd_config
}
initsetup       # initsetup fonksiyonunu çalıştırır ve kaynak dosyayı indirir
setup           # setup fonksiyonu çalışır ve derleme öncesi kaynak dosyaların ayalanması sağlanır.
build           # build fonksiyonu çalışır ve kaynak dosyaları derlenir.
package         # package fonksiyonu çalışır, yükleme öncesi ayarlamalar yapılır ve yüklenir.
```

Ek dosyaları indirmek için tıklayınız. **Paket Derleme Yöntemi** konusunda anlatıldığı gibi derleme işlemini yapınız.

pam

PAM, farklı kimlik doğrulama yöntemlerini kolayca eklemeyi ve değiştirmeyi sağlayan kütüphanedir.

Derleme

```
#-----------------------------------------------------------------------------------------------------------------------
#!/usr/bin/env bash
name="pam"
version="1.6.0"
depends="libtirpc,libxcrypt,libnsl,audit"
description="PAM (Pluggable Authentication Modules) library'"
source="https://github.com/linux-pam/linux-pam/releases/download/v$version/Linux-PAM-$version.tar.xz"
display=":$(ls /tmp/.X11-unix/* | sed 's#/tmp/.X11-unix/X##' | head -1)"  # Display adı
user=$(who | grep "(${display})" | awk '{print $1}')                      # Display kullanıcısı

ROOT="/home/$user/distro"
ROOTBUILDDIR="$ROOT/build"                # Derleme dizini
BUILDDIR="$ROOT/build/build-${name}-${version}" # Alt dizin
DESTDIR="$ROOT/rootfs"                     # Yükleme dizini
PACKAGEDIR=$(pwd)                          # Paket dizini
SOURCEDIR="$ROOT/build/${name}-${version}" # Kaynak dizini

initsetup(){
            mkdir -p  $ROOTBUILDDIR #derleme dizini yoksa oluşturuluyor
            rm -rf $ROOTBUILDDIR/* #içeriği temizleniyor
            cd $ROOTBUILDDIR #dizinine geçiyoruz
    wget ${source}
    for f in *\ *; do mv "$f" "${f// /}"; done #isimde boşluk varsa silme işlemi yapılıyor
            dowloadfile=$(ls|head -1)
            filetype=$(file -b --extension $dowloadfile|cut -d'/' -f1)
            if [ "${filetype}" == "???" ]; then unzip  ${dowloadfile}; else tar -xvf ${dowloadfile};fi
            director=$(find ./* -maxdepth 0 -type d)
            directorname=$(basename ${director})
            if [ "${directorname}" != "${name}-${version}" ]; then mv $directorname ${name}-${version};fi
            mkdir -p $BUILDDIR&&mkdir -p $DESTDIR&&cd $SOURCEDIR
}

setup(){
    ./configure --prefix=/usr --sbindir=/usr/sbin --libdir=/usr/lib64 \
        --enable-securedir=/usr/lib64/security --enable-static --enable-shared --disable-nls --disable-selinux
        }
build(){
        make
}
package(){
        make install DESTDIR=$DESTDIR
        chmod +s "$DESTDIR"/usr/sbin/unix_chkpwd
        ${DESTDIR}/sbin/ldconfig -r ${DESTDIR}          # sistem guncelleniyor
}

initsetup       # initsetup fonksiyonunu çalıştırır ve kaynak dosyayı indirir
setup           # setup fonksiyonu çalışır ve derleme öncesi kaynak dosyaların ayalanması sağlanır.
build           # build fonksiyonu çalışır ve kaynak dosyaları derlenir.
package         # package fonksiyonu çalışır, yükleme öncesi ayarlamalar yapılır ve yüklenir.
```

Paket Derleme Yöntemi konusunda anlatıldığı gibi derleme işlemini yapınız.

ISO Hazırlama

Önceki bölümlerde, gerekli paketlerin derlenmesi tamamlandı. Bu paketler, oturum açtığınız kullanıcının ev dizininde **$HOME/distro/rootfs** konumunda yer alacaktır. Burada **$HOME**, o anki kullanıcıya göre değişiklik gösterecektir. Örneğin, kullanıcı adımız **bd** ise, **$HOME** dizini **/home/bd/** olacaktır. Bu durumda, **$HOME/distro/rootfs** aslında **/home/bd/distro/rootfs** dizinini temsil eder.

ISO hazırlama süreci, yazının başında anlattığımız **Temel Linux Sistemi Oluşturma** ISO yapımına benzer bir şekilde ilerleyecektir. Oluşturulacak **iso** dosyasının temel yapısı aşağıdaki gibi olacaktır. Bu yapıyı oluşturan dört temel dosyanın her biri için hazırlık adımları tek tek açıklanacak ve en sonunda hepsini bir araya getiren bir ISO oluşturma scripti verilecektir.

```
$HOME/distro/iso/boot/grub/grub.cfg
$HOME/distro/iso/boot/initrd.img
$HOME/distro/iso/boot/vmlinuz
$HOME/distro/iso/live/filesystem.squashfs
```

1-grub.cfg Hazırlama

Sistemin açılmasını sağlayan temel dosyalardan biri **grub.cfg** dosyasıdır. Genel olarak basit bir **grub.cfg** şu şekildedir:

```
linux /boot/vmlinuz
initrd /boot/initrd.img
boot
```

Bu örnekteki sistem için **grub.cfg** dosyası aşağıdaki gibi düzenlenmiştir. Burada dikkat edilmesi gereken noktalar:

- Menü seçeneklerinde **live** ifadesinin kullanılması.
- Sistemin **openrc** ile başlatılmasını sağlamak için gerekli kernel parametrelerinin eklenmesi.
- Kurulum için özel bir **init=/bin/kur** satırı kullanılması.

İşte örnek **grub.cfg** oluşturma komutları:

```
cat << EOF > "$HOME/distro/iso/boot/grub/grub.cfg"
set timeout=3        # Timeout for menu
set default=1        # Default boot entry
set menu_color_normal=white/black    # Menu Colours
set menu_color_highlight=white/blue  # Menu Colours
insmod all_video; terminal_output console; terminal_input console

menuentry "Canli(live) GNU/Linux(kly)" --class liveiso {
 linux /boot/vmlinuz boot=live init=/sbin/openrc-init net.ifnames=0 \
     biosdevname=0
 initrd /boot/initrd.img
}
menuentry "Kur GNU/Linux(kly)" --class liveiso {
   linux /boot/vmlinuz boot=live init=/bin/kur quiet
   initrd /boot/initrd.img
   }
EOF
```

2-initrd.img Oluşturma/Güncelleme

initrd.img dosyasının güncellenmesi veya oluşturulması için aşağıdaki adımlar izlenebilir.

Varsayılan sistemde çalışıyorsanız doğrudan aşağıdaki komut yeterlidir:

```
# initrd günceller
/usr/sbin/update-initramfs -u -k $(uname -r)
```

Ancak bir **chroot** ortamında çalışıyorsanız (yani rootfs içine girip işlem yapacaksanız), önce **dev**, **sys**, **proc**, **run**, **tmp** dizinlerinin bağlanması gerekir. Bu işlemler root yetkisi ile yapılmalıdır.

Aşağıda örnek bir **initrd.img oluşturma scripti** yer almaktadır:

```
#-------------------------------------------------------------------------
rootfs="$HOME/distro/rootfs"
distro="$HOME/distro"

## Gerekli dizinler oluşturuluyor bağlanıyor
for dir in dev sys proc run tmp; do
 mkdir -p "$rootfs/$dir"
 mount --bind /$dir $rootfs/$dir
done

## Kernel versiyonu tespiti
fname=$(basename $rootfs/boot/config*)
kversion=${fname:7}
mv $rootfs/boot/config* $rootfs/boot/config-$kversion
cp $rootfs/boot/config-$kversion $rootfs/etc/kernel-config

## initramfs oluşturuluyor
chroot $rootfs update-initramfs -u -k $kversion

## Dizin bağlantıları kaldırılıyor
for dir in dev sys proc run tmp; do
     umount -lf -R $rootfs/$dir 2>/dev/null
done

## initrd.img dosyası kopyalanıyor
cp -pf $rootfs/boot/initrd.img-* $distro/iso/boot/initrd.img
```

3-vmlinuz Hazırlama

Kernel dosyamız olan **vmlinuz**, ISO dizinine aşağıdaki komutla kopyalanır:

```
#-------------------------------------------------------------------------
rootfs="$HOME/distro/rootfs"
distro="$HOME/distro"

# Kernel dosyasını kopyala
cp -pf $rootfs/boot/vmlinuz-* $distro/iso/boot/vmlinuz

# Sistemde rootfs dışında boot bölümü olduğu için boot dizini silinebilir
#rm -rf $rootfs/boot
```

4-filesystem.squashfs Hazırlama

Sistemin **live** olarak çalıştırılabilmesi ve kurulum için kaynak oluşturacak **filesystem.squashfs** dosyası oluşturulur:

```
#-------------------------------------------------------------------------
cd $HOME/distro/
mksquashfs $HOME/distro/rootfs $HOME/distro/filesystem.squashfs -comp xz -wildcards
mv $HOME/distro/filesystem.squashfs $HOME/distro/iso/live/filesystem.squashfs
```

5-ISO Dosyasının Oluşturulması

Tüm dosyalar konumlarına uygun hazırlandıktan sonra, **grub-mkrescue** aracı ile ISO dosyamızı oluşturabiliriz:

```
#-------------------------------------------------------------------------
cd $HOME/distro
grub-mkrescue iso/ -o kly.iso
```

6-ISO'nun Test Edilmesi

Oluşturduğumuz ISO dosyasını **qemu** veya **VirtualBox** ile test edebiliriz. Linux üzerinde terminalden **qemu** kullanarak aşağıdaki şekilde test gerçekleştirebilirsiniz.

```
# qemu ile ISO'nun test edilmesi
qemu-system-x86_64 -cdrom $HOME/distro/kly.iso -m 1G
```

ISO Oluşturma Scripti

Bu dokümanda anlatılan paketleri kurulduğu **/home/$user/distro/rootfs** dizini kullanarak, **/home/$user/distro/iso/** konumuna **kly.iso** dosyasını üretir. Bu bölümde anlatılan tüm aşamalar tek script halinde aşağıda verilmiştir. Bu scripti **sudo** yetkisiyle çalıştırınız.

```
#!/bin/bash
#-----------------------------------------------------------------------------------------------------------------------
tepmPath="${PATH}"
export PATH=$PATH:/sbin/
display=":$(ls /tmp/.X11-unix/* | sed 's#/tmp/.X11-unix/X##' | head -n 1)"
user=$(who | grep '('$display')' | awk '{print $1}') # Kullanıcı&ekran bul
distro="/home/$user/distro"; rootfs="$distro/rootfs"; rm -rf "$distro/iso"

echo "[+] $rootfs dizininin izinleri düzenleniyor..."
chown -R root:root ${rootfs}/bin ${rootfs}/sbin ${rootfs}/usr/bin ${rootfs}/usr/sbin #sahibi root:root olsun
chown -R root:root ${rootfs}/var ${rootfs}/run ${rootfs}/root   #sahibi root:root olsun
find ${rootfs} -type d -exec chmod 755 {} \;    # Tüm dizinlere 755 ver
chmod 600 ${rootfs}/etc/shadow etc/gshadow 2>/dev/null  # Hassas dosyaları ayarla
chmod 1777 ${rootfs}/tmp # /tmp için sticky bit
echo "[+] $rootfs dizininden iso oluşturuluyor..."      # chroot ortamı için gerekli bind işlemleri
for dir in dev dev/pts proc sys; do mount -o bind /$dir $rootfs/$dir; done

# root ve live kullanıcı eklenmesi
chroot $rootfs echo -e "1\n1\n"|chroot $rootfs passwd root
chroot $rootfs userdel -r live
chroot $rootfs useradd live -m -s /bin/sh -d /home/live
chroot $rootfs echo -e "live\nlive\n"|chroot $rootfs passwd live
# /root ve /home/live için sahiplik ve izinleri ayarlanıyor
chown live:live ${rootfs}/home/live
chmod 700 ${rootfs}/home/live
chown root:root ${rootfs}/root
chmod 700 ${rootfs}/root
for grp in users tty wheel cdrom audio dip video plugdev netdev; do
chroot $rootfs usermod -aG $grp live || true
done
# agetty ayarları
sed -i "/agetty_options/d" $rootfs/etc/conf.d/agetty
echo -e "\nagetty_options=\"-l /usr/bin/login\"" >> $rootfs/etc/conf.d/agetty
# initrd güncellemesi
fname=$(basename $rootfs/boot/config*)
kversion=${fname:7}
mv $rootfs/boot/config* $rootfs/boot/config-$kversion
cp $rootfs/boot/config-$kversion $rootfs/etc/kernel-config
chroot $rootfs update-initramfs -u -k $kversion
# chroot bind dizinlerini ayır
for dir in dev dev/pts proc sys ; do
while umount -lf -R $rootfs/$dir 2>/dev/null ; do true; done
done
# ISO dizin yapısını oluştur
mkdir -p "$distro/iso/boot/grub" "$distro/iso/live"
cp -pf $rootfs/boot/initrd.img-* $distro/iso/boot/initrd.img # initrd.img
cp -pf $rootfs/boot/vmlinuz-* $distro/iso/boot/vmlinuz # vmlinuz kopyala
mksquashfs $rootfs $distro/filesystem.squashfs -comp xz -wildcards # squashfs
mv $distro/filesystem.squashfs $distro/iso/live/filesystem.squashfs
# grub.cfg dosyasını yaz
cat << EOF > "$distro/iso/boot/grub/grub.cfg"
set timeout=6; set default=1; terminal_input console;
menuentry "Canli(live) GNU/Linux 64-bit" --class liveiso {
linux /boot/vmlinuz boot=live init=/sbin/openrc-init net.ifnames=0 \
biosdevname=0
initrd /boot/initrd.img
}
menuentry "Kur GNU/Linux 64-bit" --class liveiso {
linux /boot/vmlinuz boot=live init=/bin/kur quiet
initrd /boot/initrd.img
}
EOF
grub-mkrescue $distro/iso/ -o $distro/kly.iso   #iso oluşturuluyor
export PATH="${tempPath}"
```

Kaynaklar:

```
- https://www.subrat.info/build-kernel-and-userspace/ - 08/07/2025
- https://medium.com/@chienhaotan/compiling-and-running-a-minimal-kernel-with-busybox-bfc45a991017 - 08/07/2025
- https://wiki.archlinux.org/title/GRUB_(T%C3%BCrk%C3%A7e)  - 08/07/2025
- https://chatgpt.com/share/6875084c-6050-8012-9229-a37b47351aa2 - 14/07/2025
```

Oluşan Sistemin Çalıştırılması İncelenmesi

Yaptığımız sistem ISO haline dönüştürüldü ve artık **test etme** ile **inceleme** aşamasına geçtik. ISO hazırlama bölümünde aşağıdaki komutla ISO'yu oluşturduk:

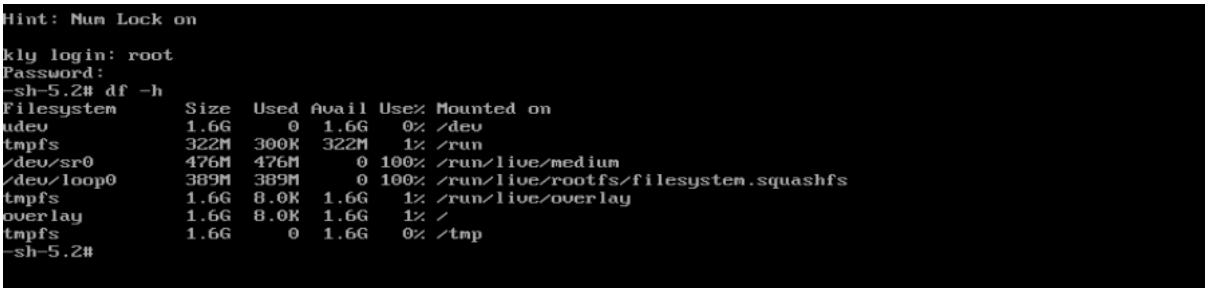
```
grub-mkrescue $distro/iso/ -o $distro/kly.iso
```

Oluşturulan ISO dosyası `$distro/kly.iso` konumunda yer almakta olup, bu konum `$HOME/distro/kly.iso` dizinine karşılık gelmektedir.

Bu dokümanda anlatılan paketlerin **kly Paket Sistemiyle** hazırlanmış **Temel Sistem** isosu hazırlandı. İsoyu https://github.com/kendilinuxunuyap/kly-base-distro/releases/download/current/kly-base-distro.iso adresinden indirebilirsiniz.

Şimdi hazırlanan ISO'yu **QEMU** veya **VirtualBox** kullanarak çalıştıralım. Ekran görüntüleri aşağıda verilmiştir.

Canlı(live) Sistem Kullanımı



Canlı(live) sistem çalıştırıldığında overlay live bir sistemin açıldığını görmekteyiz. Canlı(live) sistemde kullanıcı adları ve parolaları;

- Kullanıcı: root Parola: 1
- Kullanıcı: live Parola: live

Sistem Kurulumu

Hazırlanan ISO ile birlikte, farklı kurulum araçları da gelebilir. Bu araçlar, çeşitli kurulum yöntemlerini destekleyebilir. En sık kullanılan kurulum yöntemleri şunlardır:

1. Tek bölüme sistem kurulumu
2. UEFI sistem kurulumu (boot + sistem)

Bu bölümde, tek bölüm kurulum ve boot + sistem şeklinde iki farklı kurulum yöntemi sırayla anlatılacaktır. Anlatılan yöntemler, farklı kullanıcı senaryolarına cevap verebilecek şekilde tasarlanmıştır.

Ancak dikkat edilmesi gereken önemli bir nokta şudur: Her yöntemi ayrı ayrı son kullanıcıya seçenek olarak sunmak, özellikle tecrübesiz kullanıcılar için kafa karıştırıcı olabilir. Örneğin:

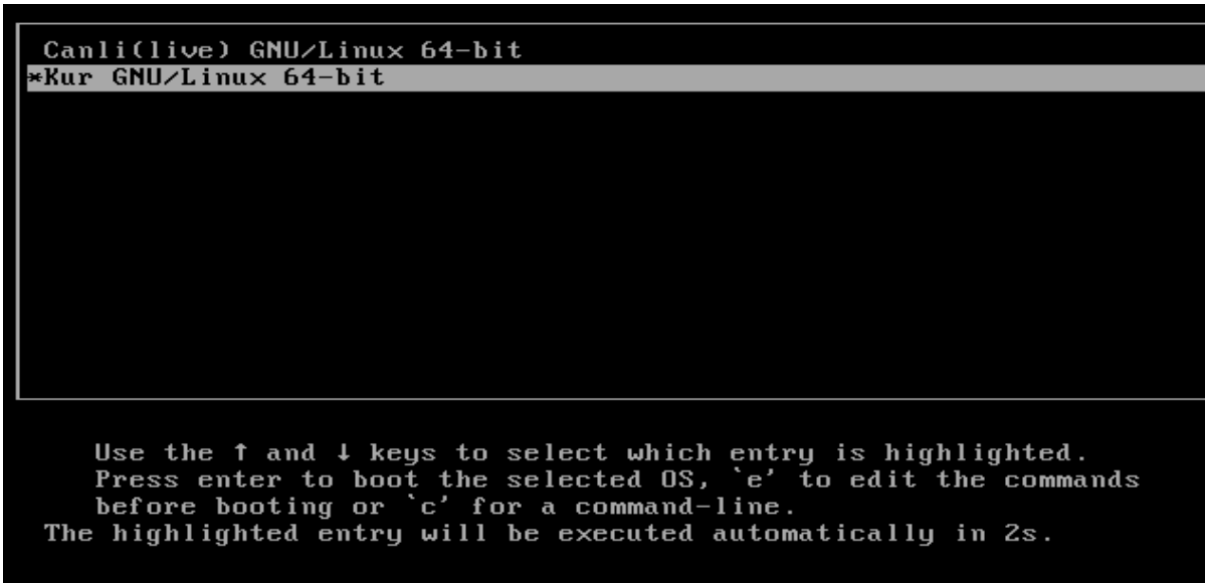
- Kullanıcı, kurulum sırasında **EFI** seçeneğini işaretlediğinde fakat sistem aslında **Legacy BIOS** ise, ya da tam tersi durumda, yanlış seçim yapabilir.
- Ayrıca, **sda** ve **nvme** diskler arasında farklı kurulum senaryoları gerekebilir.

Bu nedenle hazırlanan kurulum sistemi, aşağıdaki tüm olası senaryolara otomatik cevap verecek şekilde tasarlanmıştır:

- **Legacy BIOS → sd*** disk tipi
- **Legacy BIOS → nvme*** disk tipi (**desteklenmemektedir**)
- **UEFI → sd*** disk tipi
- **UEFI → nvme*** disk tipi

> Note
>
> Bu senaryolara göre hazırlanan kurulum scripti, **dialog** aracı kullanılarak oluşturulmuştur. İlgili kurulum scriptleri, **base-file** paketinin içindeki **files.rar** arşivinde yer almaktadır.
>
> Bu bölümde, sadece **Legacy BIOS** ve **UEFI** sistem kurulumunun genel adımları anlatılacaktır. Bu yöntemleri kendi ihtiyaçlarınıza göre düzenleyerek kullanabilirsiniz.

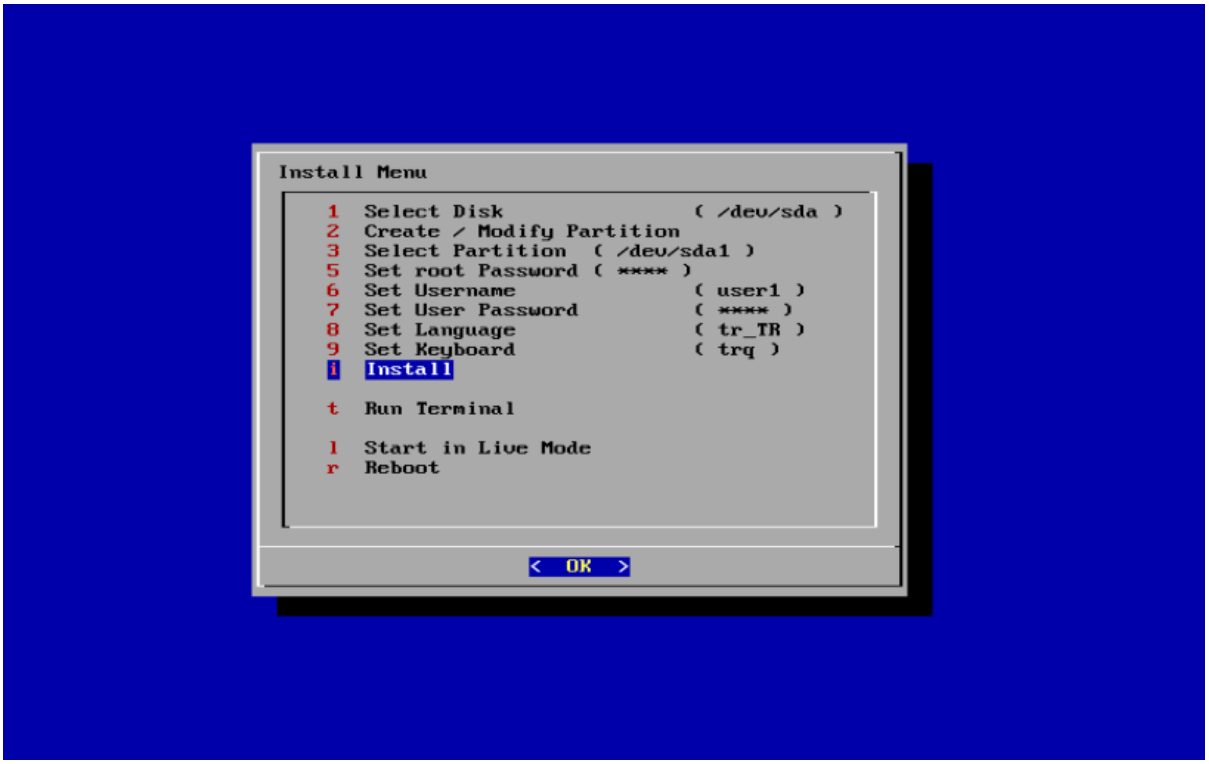
Sistemin kurulumu için resimlerde görünen sıraya göre seçimler yapmalıyız.



Kurulum menüsünde kullanıcı adları ve parolaları, klayve varsayılan olarak;

- Kullanıcı: root Parola: 1
- Kullanıcı: user1 Parola: 1
- Dil : tr_TR
- Klavye : trq

menüden değişiklik yapabilirsiniz. Değişiklik yapmadan sadece kurulum diskini ve disk bölümünü seçip Install(Yükle) işlemi yapabilirsiniz.

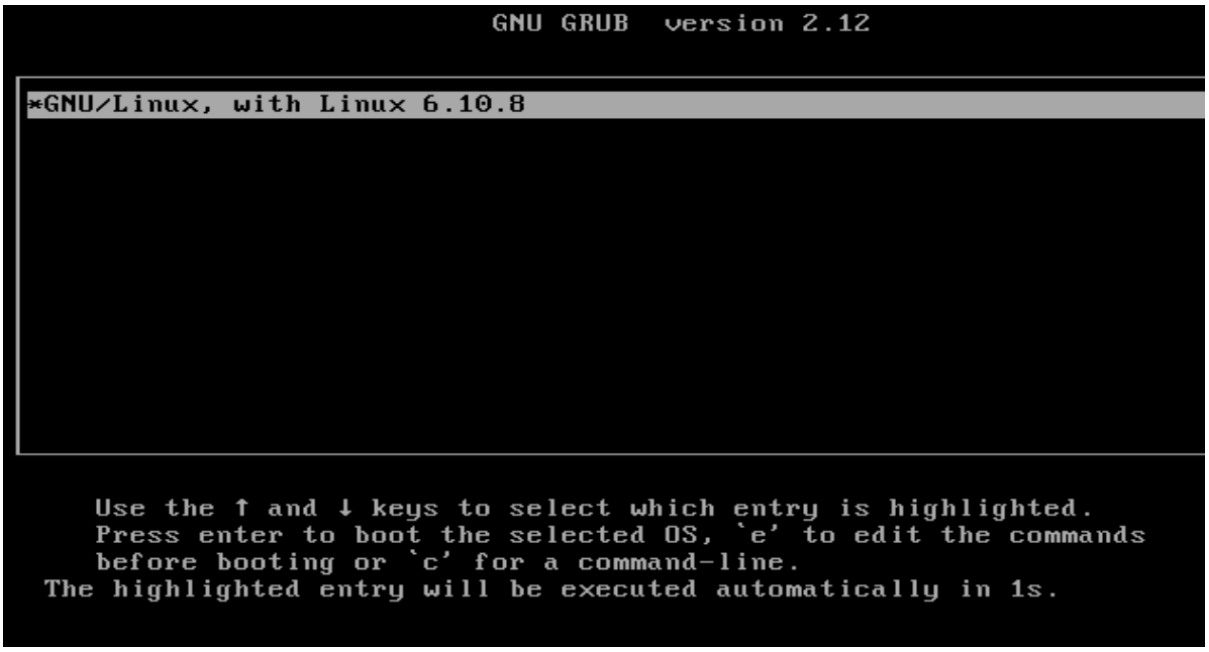


kly [Çalışıyor] - Oracle VM VirtualBox

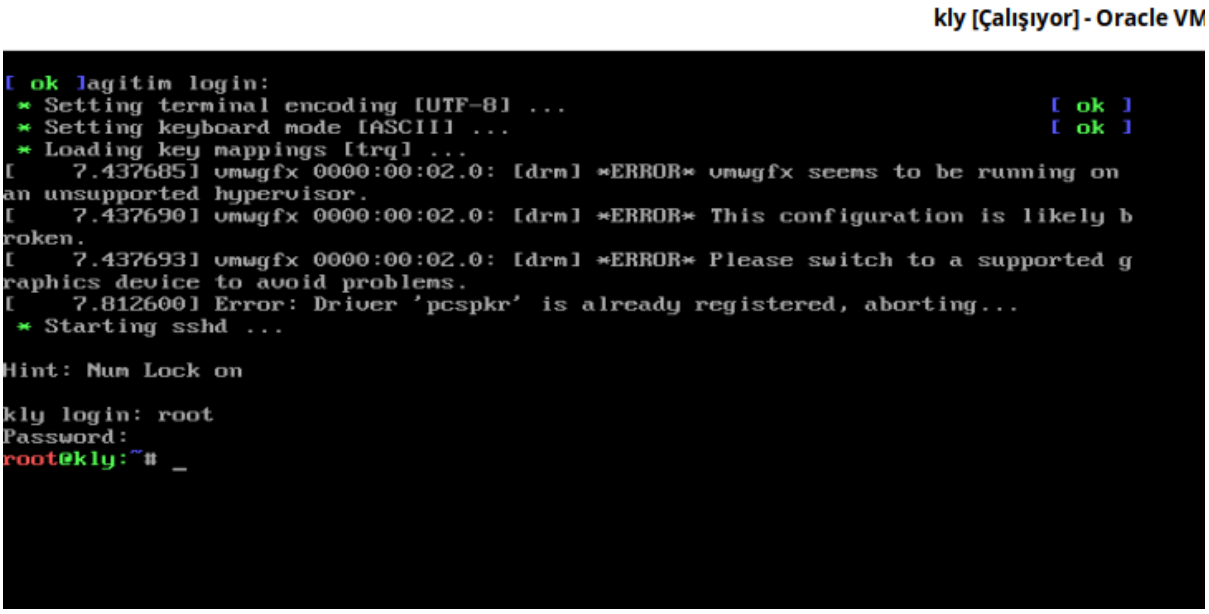
```
kuruluma geçildi........
legacy kurulum ........
Kurulum Yap■lacak Disk sda
mount: /kaynak: WARNING: source write-protected, mounted read-only.
**********disk bölümleri biçimlendiriliyor **********
e2fsck 1.47.0 (5-Feb-2023)
e2fsck: need terminal for interactive repairs
tune2fs 1.47.0 (5-Feb-2023)
Recovering journal.

This operation requires a freshly checked filesystem.

Please run e2fsck -f on the filesystem.

mke2fs 1.47.0 (5-Feb-2023)
Creating filesystem with 2096896 4k blocks and 524288 inodes
Filesystem UUID: 9ea082d0-a034-4dab-8e2f-564e68c99c9c
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632

Allocating group tables: done
Writing inode tables: done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information: done

Information: You may need to update /etc/fstab.

********** kurulum ba■lad■**********
Sistem yuklenmeye baslandi. Tahmini 3-5 dakika surecektir.. Lutfen bekleyiniz.....
.....
_
```

Sistemin Çalışması

Sistem kurulumu gerçekleştiğinde sistem resimde görüldüğü gibi açılmalıdır.



Sisteme **root** kullanıcısı olarak giriş yapıldığı görülmektedir.



Oluşan Sistemin Değerlendirmesi

Mevcut sistem **Debian** ortamında derlenmiştir. Paketler derlenirken, Debian'a özgü fakat bizim sistemimiz için gerekli olmayan bazı ayarlar ve bağımlılıklar da yeni sisteme taşınmış olabilir. Bu tür taşınan unsurları ortadan kaldırmak için, tüm paketlerin tamamen kendi sistemimiz üzerinde yeniden derlenmesi gereklidir.

İlk derleme süreci Debian üzerinde yapılmış olsa da, sonraki derleme adımı mutlaka hazırladığımız yeni sistemde gerçekleştirilmelidir. Bu yöntem, sistemdeki eksiklikleri tespit etmek ve gözden kaçan bağımlılıkları ortaya çıkarmak açısından kritik öneme sahiptir.

Derleme işlemimin yapılabilmesi için oluşturulan sistemdeki paketlere ek paketler derlenmesi gerekmektedir. Bu paketlerin en önemlileri **gcc, binutils, mpfr, libmpc, zlib, libisl, make**'dir.

Bu dokümanın amacı, bir sistemin nasıl derlenip çalıştırılacağını adım adım bir rehber olarak sunmaktır. Bundan sonraki aşamada ise, tüm paketlerin yeni sistem üzerinde eksiksiz bir şekilde derlenmesi hedeflenmelidir.

Tüm paketler yeni sistemde başarıyla derlendikten ve sistem sorunsuz şekilde çalıştıktan sonra, **x11** ortamını derleme aşamasına geçilebilir. Dokümanda yer alan paket derleme betikleri kullanılarak **x11** kolaylıkla derlenebilir. Bu noktaya kadar sorunsuz ilerlenmişse, **x11**'in derlenmesi yalnızca zaman alacak bir işlemdir.

x11'in en temel paketi **xorg-server, mesa, llvm, cairo** paketleridir. Bu paketlere aşağıdaki adreslerden ulaşılabilir.

1. xorg-server: https://www.x.org/releases/individual/xserver/
2. mesa : https://gitlab.freedesktop.org/mesa/mesa/-/tags
3. llvm : https://github.com/llvm/llvm-project/tags
4. cairo: https://gitlab.freedesktop.org/cairo/cairo/-/tags

Tüm paketleri derlesek bile **xorg-server, mesa, llvm, cairo** paketleri düzgün ve uyumlu versiyonları olmadığı zaman **x** penceremiz açılmayacaktır.

Buradaki tüm paketler ve bağımlılıkları derlendikten sonra **Xorg:0** şeklinde elle çalıştırarak hata ayıklama yapılmalıdır.

Bu dokümanın devamı niteliğinde temel bir **x11** ortamı nasıl derleneceğini anlatan bir doküman hazırlamayı planlamaktayız.

Paket Sistemi Tasarlama

Paket Sitemi

Paket sistemi dağıtımların paketleri yükleme, kaldırma, güncelleme gibi temel işlemlerin yapılmasını sağlayan en önemli bileşenidir.

Paket sistemleri bir paketi yüklemek istediğinde, genellikle daha önceden derlenmiş paketleri veya yükleme aşamasında derleyebilir. Önceden derlenmiş olan yapıya(binary==ikili), yükleme aşamasında derleme işleminede (source==kaynak) paket sistemi denir.

Gnu/Linux deneyimi az olan kullanıcılar için daha önceden hazırlanmış ikili paketler tercih edilmektedir.

Dağıtımlarda uygulamalar paketler halinde hazırlanır. Bu paketleri dağıtımda kullanabilmek için temel işlemler şunlardır;

1. Paket Oluşturma
2. Paket Liste İndexi Güncelleme
3. Paket Kurma
4. Paket Kaldırma
5. Paket Yükseltme gibi işlemleri yapan uygulamaların tamamı paket sistemi olarak adlandırılır.

Paket sisteminde, uygulama paketi haline getirilip sisteme kurulur. Genelde paket sistemi dağıtımın temel bir parçası olması sebebiyle üzerinde yüklü gelir.

Bazı dağıtımların kullandığı paket sistemleri şunlardır.

- apt: Debian dağıtımının kullandığı paket sistemi.
- emerge :Gentoo dağıtımının kullandığı paket sistemi.

Not: Bu dokümanda **Paket Oluşturma, Paket Liste İndexi Güncelleme, Paket Kurma, Paket Kaldırma** işlemini yapan scriptleri tasarlanmaktadır.

kly Paket Sistemi

Bu dokümanda hazırlanan dağıtımın paket sistemi için ise **kly(KendiLinuxunuYap)** olarak ifade edeceğimiz paket sistemi adını kullandık. kly paket sistemindeki beş temel işlemin nasıl yapılacağı ayrı başlıklar altında anlatılacaktır. Paket sistemi derlemeli bir dil yerine bash script ile yapılacaktır. Bu dokumanı takip eden kişi, bu dokümanda yazılanları anlaması için orta seviye bash script bilmesi gerekmektedir.

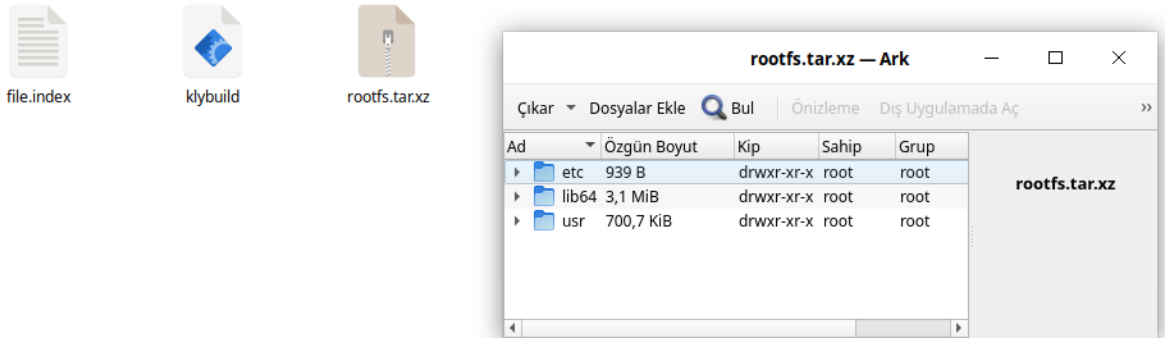
Paket Oluşturma

Paket sisteminde en önemli kısımlardan birisi paket oluşturmadır. Bu işlem paketin derlenmesi ve derlenmiş paketin belirli bir yapıyla saklanması olayıdır. Bu saklanan paket daha sonra ihtiyaç halinde uzaktan(internet üzerinden) ve yerelden istediğimiz sisteme kurma işlemidir. Bu başlıkta paketin derlenmesi ve saklanması(paket oluşturma) anlatılacaktır.

Paket oluşturma işlemi sırayla şu aşamalardan oluşmaktadır.

1. Paketin indirilmesi
2. Paketin derleme öncesi hazırlanması(configure)
3. Paketin derlenmesi
4. Derlenmiş paketin bir dizine yüklenmesi
5. Yüklenen dizindeki dosya ve dizin yapısının konum listesini tutan file.index oluşturulması
6. Derlenmiş paketin bir dizinin sıkıştırılması
7. Sıkıştırılmış derlenmiş dizin, file.index ve derleme talimatının paket isim ve versiyonuyla tekrardan sıkıştırılması

Burada maddeler halinde anlatılan işlem adımlarını bir paket oluşturma amacıyla sırasıyla yapmamız gerekmektedir. 7. maddede anlatılan son sıkıştırılma öncesi yapı aşağıda gösterilmiştir.



kly Paket Oluşturma

kly paket sisteminin temel parçalarından en önemlisi paket oluşturma uygulamasıdır. Dokümanda temel paketlerin nasıl derlendiği **Paket Derleme** başlığı altında anlatılmıştı. Bir paket üzerinden(readline) örneklendirerek paketimizi oluşturacak scriptimizi yazalım.

Dokümanda readline paketi nasıl derleneceği aşağıdaki script olarak verilmiştir.

```
#!/usr/bin/env bash
version="8.2"
name="readline"
depends="glibc"
description="readline kütüphanesi"
source="https://ftp.gnu.org/pub/gnu/readline/${name}-${version}.tar.gz"
groups="sys.apps"

display=":$(ls /tmp/.X11-unix/* | sed 's#/tmp/.X11-unix/X##' | head -n 1)"      #Detect the name of the display in use
user=$(who | grep '('$display')' | awk '{print $1}')    #Detect the user using such display
ROOTBUILDDIR="/home/$user/distro/build" # Derleme konumu
BUILDDIR="/home/$user/distro/build/build-${name}-${version}" #Derleme yapılan paketin derleme konumun
DESTDIR="/home/$user/distro/rootfs" #Paketin yükleneceği sistem konumu
PACKAGEDIR=$(pwd) #paketin derleme talimatının verildiği konum
SOURCEDIR="/home/$user/distro/build/${name}-${version}" #Paketin kaynak kodlarının olduğu konum

initsetup(){
            mkdir -p  $ROOTBUILDDIR #derleme dizini yoksa oluşturuluyor
            rm -rf $ROOTBUILDDIR/* #içeriği temizleniyor
            cd $ROOTBUILDDIR #dizinine geçiyoruz
    wget ${source}
    for f in *\ *; do mv "$f" "${f// /}"; done #isimde boşluk varsa silme işlemi yapılıyor
            dowloadfile=$(ls|head -1)
            filetype=$(file -b --extension $dowloadfile|cut -d'/' -f1)
            if [ "${filetype}" == "???" ]; then unzip  ${dowloadfile}; else tar -xvf ${dowloadfile};fi
            director=$(find ./* -maxdepth 0 -type d)
            directorname=$(basename ${director})
            if [ "${directorname}" != "${name}-${version}" ]; then mv $directorname ${name}-${version};fi
            mkdir -p $BUILDDIR&&mkdir -p $DESTDIR&&cd $SOURCEDIR
}

setup(){
        cp -prvf $PACKAGEDIR/files $SOURCEDIR/
        ./configure --prefix=/usr \
                --libdir=/usr/lib64
}

build(){
        make SHLIB_LIBS="-L/tools/lib -lncursesw"
}

package(){
        make SHLIB_LIBS="-L/tools/lib -lncursesw" DESTDIR="$DESTDIR" install pkgconfigdir="/usr/lib64/pkgconfig"

        install -Dm644 $SOURCEDIR/files/inputrc "$DESTDIR"/etc/inputrc
        ${DESTDIR}/sbin/ldconfig -r ${DESTDIR}           # sistem guncelleniyor
}
initsetup       # initsetup fonksiyonunu çalıştırır ve kaynak dosyayı indirir
setup           # setup fonksiyonu çalışır ve derleme öncesi kaynak dosyaların ayalanması sağlanır.
build           # build fonksiyonu çalışır ve kaynak dosyaları derlenir.
package         # package fonksiyonu çalışır, yükleme öncesi ayarlamalar yapılır ve yüklenir.
```

Bu script readline kodunu internetten indirip derliyor ve kurulumu yapıyor. Aslında bu scriptle **paketleme**, **paket kurma** işlemini bir arada yapıyor. Bu işlem mantıklı gibi olsada paket sayısı arttıkça ve rutin yapılan işlemleri tekrar tekrar yapmak gibi işlem fazlalığına sebep olmaktadır.

Bu sebeplerden dolayı **readline** paketleme scriptini yeniden düzenleyelim. Yeni düzenlenen halini **klypaketle** ve **klybuild** adlı script dosyaları olarak düzenleyeceğiz. Genel yapısı aşağıdaki gibi olacaktır. Devamında ise **packageindex** ve **packagecompress** fonksiyonları klypaketle dosyasına eklenecektir.

klybuild Dosyası

```
setup() {}
build() {}
package() {}
```

klypaketle Dosyası

```
#genel değişkenler tanımlanır
initsetup() {}

#klybuild dosya fonksiyonları birleştiriliyor
source klybuild # bu komutla setup build package fonsiyonları klybuild doyasından alınıp birleştiriliyor

packageindex() {}
packagecompress() {}
```

Aslında yukarıdaki **klypaketle** ve **klybuild** adlı script dosyaları tek bir script dosyası olarak **klypaketle** dosyası. İki dosyayı birleştiren **source klybuild** komutudur. **klypaketle** dosyası aşağıdaki gibi düşünebiliriz.

```
#genel değişkenler tanımlanır
initsetup() {}

setup() {} #klybuild dosyasından gelen fonksiyon, "source klybuild" komutu sonucu gelen fonksiyon
build() {} #klybuild dosyasından gelen fonksiyon, "source klybuild" komutu sonucu gelen fonksiyon
package() {} #klybuild dosyasından gelen fonksiyon, "source klybuild" komutu sonucu gelen fonksiyon

packageindex() {}
packagecompress() {}
```

Bu şekilde ayrılmasının temel sebebi **klypaketle** scriptinde hep aynı işlemler yapılırken **klybuild** scriptindekiler her pakete göre değişmektedir. Böylece paket yapmak için ilgili pakete özel **klybuild** dosyası düzenlememiz yeterli olacaktır. **klypaketle** dosyamızda **klybuild** scriptini kendisiyle birleştirip paketleme yapacaktır.

klybuild Dosyamızın Son Hali

```
#!/usr/bin/env bash
version="8.2"
name="readline"
depends="glibc"
description="readline kütüphanesi"
source="https://ftp.gnu.org/pub/gnu/readline/${name}-${version}.tar.gz"
groups="sys.apps"
#2. madde, derleme öncesi hazırlık
setup(){
        cp -prvf $PACKAGEDIR/files $BUILDDIR/
        $SOURCEDIR/configure --prefix=/usr \
                --libdir=/usr/lib64
}
#3. madde, paketin derlenmesi
build(){
        make SHLIB_LIBS="-L/tools/lib -lncursesw"
}
#4. madde, derlenen paketin bir dizine yüklenmesi
package(){
        make SHLIB_LIBS="-L/tools/lib -lncursesw" DESTDIR="$DESTDIR" install pkgconfigdir="/usr/lib64/pkgconfig"

        install -Dm644 files/inputrc "$DESTDIR"/etc/inputrc
}
```

klypaketle Dosyamızın Son Hali

```
#!/usr/bin/env bash
set -e
paket=$1
dizin=$(pwd)
echo "Paket : $paket"
source ${paket}/klybuild
ROOTBUILDDIR="/tmp/kly/build"
BUILDDIR="/tmp/kly/build/build-${name}-${version}" #Derleme yapılan dizin
DESTDIR="/tmp/kly/build/rootfs-${name}-${version}" #Paketin yükleneceği sistem konumu
PACKAGEDIR="$dizin/$paket"
SOURCEDIR="/tmp/kly/build/${name}-${version}"
# 1. madde, paketin indirilmesi
initsetup(){
        mkdir -p $ROOTBUILDDIR #derleme dizini yoksa oluşturuluyor
        rm -rf $ROOTBUILDDIR/* #içeriği temizleniyor
        cd $ROOTBUILDDIR #dizinine geçiyoruz
        if [ -n "${source}" ]
        then
        wget ${source}
        dowloadfile=$(ls|head -1)
        filetype=$(file -b --extension $dowloadfile|cut -d'/' -f1)
        if [ "${filetype}" == "???" ]; then unzip ${dowloadfile}; else tar -xvf ${dowloadfile};fi
        director=$(find ./* -maxdepth 0 -type d)
        directorname=$(basename ${director})
        if [ "${directorname}" != "${name}-${version}" ]; then mv $directorname ${name}-${version};fi
        fi
        mkdir -p $BUILDDIR&&mkdir -p $DESTDIR&&cd $BUILDDIR
        cp $PACKAGEDIR/klybuild $ROOTBUILDDIR/
}
# 6. madde, paketlenecek dosların listesini tutan file.index dosyası oluşturulur
packageindex()
{
        rm -rf file.index
        cd /tmp/kly/build/rootfs-${name}-${version}
        find . -type f | while IFS= read file_name; do
        if [ -f ${file_name} ]; then echo ${file_name:1}>>../file.index; fi
        done
        find . -type l | while IFS= read file_name; do
        if [ -L ${file_name} ]; then echo ${file_name:1}>>../file.index; fi
        done
}
# paket dosyası oluşturulur;
# rootfs.tar.xz, file.index ve klybuild dosyaları tar.gz dosyası olarak hazırlanıyor.
# 7. madde, tar.gz dosyası olarak hazırlanan dosya kly ismiyle değiştirilip paketimiz hazırlanır.
packagecompress()
{
        cd /tmp/kly/build/rootfs-${name}-${version}
        tar -cf ../rootfs.tar ./*
        cd /tmp/kly/build/
        xz -9 rootfs.tar
        tar -cvzf paket-${name}-${version}.tar.gz rootfs.tar.xz file.index klybuild
        cp paket-${name}-${version}.tar.gz ${dizin}/${paket}/${name}-${version}.kly
}
# fonksiyonlar aşağıdaki sırayla çalışacaktır.
initsetup #bu dosya içindeki fonksiyon (indirilmesi)
setup #klybuild dosyasından gelen fonksiyon (derleme öncesi hazırlık)
build #klybuild dosyasından gelen fonksiyon (derleme)
package #klybuild dosyasından gelen fonksiyon (derlenen paketin dizine yüklenemesi)
packageindex #bu dosya içindeki fonksiyon (dizine yüklelen paketin indexlenmesi)
packagecompress #bu dosya içindeki fonksiyon (index.lst, derleme talimatı ve dizinin sıkıştırılmas)
```

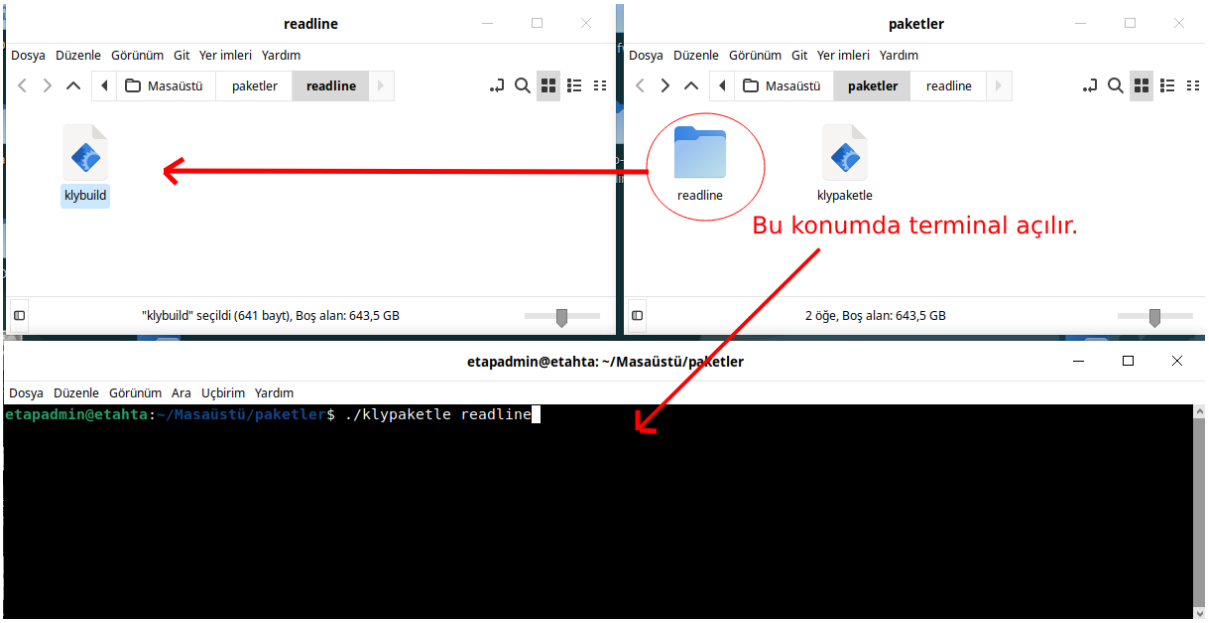
Burada **readline** paketini örnek alarak **klypaketle** dosyasının ve **klybuild** dosyasının nasıl hazırlandığı anlatıldı. Diğer paketler için sadece hazırlanacak pakete uygun şekilde **klybuild** dosyası hazırlayacağız. **klypaketle** dosyamızda değişiklik yapmayacağız. Artık **klypaketle** dosyası paketimizi oluşturan script **klybuild** ise hazırlanacak paketin bilgilerini bulunduran script dosyasıdır.

Paket Yapma

Bu bilgilere göre readline paketi nasıl oluşturulur onu görelim. Paketlerimizi oluşturacağımız bir dizin oluşturarak aşağıdaki işlemleri yapalım. Burada yine **readline** paketi anlatılacaktır.

```
mkdir readline
cd readline
# readline için hazırlanan klybuild dosyası, readline dizininin içine kopyalayın
cd ..
# klypaketle dosyamıza parametre olarak readline dizini verilmiştir.
./klypaketle readline
```

Komut çalışınca readline/readline-8.1.kly dosyası oluşacaktır. Aşağıda resimde nasıl yapıldığı gösterilmiştir. Burada anlatılan **klypaketle** script dosyasını **/bin/** konumuna oluşturnuz ve **chmod 755 /bin/klypaketle** komutuyla çalıştırma izni vermeliyiz. **kly** paket sistemi için yapılacak olan **bsppaketle, klyupdate, klykur, klykaldir** scriptlerinide **/bin/** konumunda oluşturulmalı veya kopyalanmalı ve çalıştırma izni verilmeli.



Artık sisteme kurulum için ikili dosya, kütüphaneleri ve dizinleri barındıran paketimiz oluşturuldu. Bu paketi sistemimize nasıl kurarız? konusu **Paket Kurma** başlığı altında anlatılacaktır.

Depo indexleme

Depo, paketlerimizin olduğu alandır. Depoda ne kadar paket varsa bunların isimleri sürüm numaraları gibi bilgiler ile adreslerini liste halinde oluşturma işlemine **depo indexleme** denir. Depo indexlenirken genellikle bilgiler **paket derleme talimatı(klybuild)** dosyasından alınır. Paketlerin listesi, paketler kurulurken, silinirken ve güncellenirken kullanılmaktadır.

kly github Depo Yapma

Bu doküman kullanılarak hazırlanan paketleri bilgisayarınızda bir dizinde tutabiliriz. Fakat bu çok kısıtlı bir sistem olmasına sebep olacaktır. Paketleri bir internet ortamında bir yerde saklayarak, kurmak istediğimizde internet(uzak) üzerinden kurulması daha doğru bir yöntemdir. Bu dağıtımda paketlerimizi github.com üzerinde oluşturulan bir repository üzerinden çekilmektedir. İnternetteki paketlerimizin listesi her yeni paketi yükleme sırasında güncellenmektedir. Bu işlem github hesabı üzerinden yapılmaktadır. github hakkında temel işlemler için github konusunu okuyunuz.

github üzerinde depolamak için;

- github hesabı açılır(kendilinuxunuyap)
- github repository oluşturulur(kly-binary-packages)
- kly-binary-packages deposuna aşağıda verilen index dosyasını oluşturunuz.
- kly-binary-packages deposuna .github/workflows dizinini oluşturarak aşağıda verilen **main.yml** dosyasını oluşturunuz. **main.yml** dosyasısdaki **sh index** satırı **index** scriptimizi her githuba paket gönderdiğimizde(commit) çalışacak ve **index.lst** dosyasını oluşturacaktır.
- internet üzerinden kly-binary-packages reposunda settings->action->general->Workflow permissions->Read and write permissions işaretlenmelidir.
- Yapılan paketler github üzerinde gönderilmelidir.

main.yml

```
#-------------------------------------------------------------------------------------------------------------
name: CI

on:
  push:
        branches: [ master ]
  schedule:
        - cron: "0 0 1 2 6"

jobs:
        compile:
            name: depoindex
            runs-on: ubuntu-latest
            steps:
              - name: Check out the repo
                uses: actions/checkout@v2
              - name: Run the build process with Docker
                uses: addnab/docker-run-action@v3
                with:
                    image: debian:testing
                    options: -v ${{ github.workspace }}:/root -v /output:/output
                    run: |
                        cd /root
                        sh index
              - uses: "marvinpinto/action-automatic-releases@latest"
                with:
                    repo_token: "${{ secrets.GITHUB_TOKEN }}"
                    automatic_release_tag: "current"
                    prerelease: false
                    title: "Latest release"
                    files: |
                      /output/*
```

index Dosyası

Aşağıdaki script kly paket dosyalarımızı olduğu dizinde tek tek açarak içerisinden **klybuild** dosyalarını çıkartır. Paketle ilgili bilgileri alıp **index.lst** dosyası oluşturmaktadır. İstersek paketler local ortamdada index oluşturabiliriz. Bu dokümanda github üzerinde oluşturacak şekilde anlatılmıştır. Paket indeksi oluşturan **index.lst** dosyası aşağıdaki gibi olacaktır. Listede name, version ve depends(bağımlı olduğu paketler) bilgileri bulunmaktadır. Bilgilerin arasında **|** karekteri kullanılmıştır.

```
#!/bin/sh
#set -ex
mkdir /output -p
mkdir -p /klysource
>index.lst
find * -type f -name *.kly |
                while IFS= read file_name; do
                        dosya="$(dirname $file_name)/klybuild"
                        version=$(cat $dosya|grep version=)
                        name=$(cat $dosya|grep name=)
                        depends=$(cat $dosya|grep depends=)
                        echo "$name|$version|$depends|$(dirname $file_name)">>index.lst
                done
cp -rf index.lst /output

# ****************************source files*******************************
cp -prfv ./* /klysource/

find /klysource/* -type f -name *.kly |
                while IFS= read file_name; do
                rm -rf "$file_name"
                done
tar -cf /output/klysourcepackage.tar /klysource/
rm -rf /klysource
```

index.lst İçeriği

https://github.com/kendilinuxunuyap/kly-binary-packages/releases/download/current/index.lst adresinde bulunan dosya aşağıdaki gibi liste oluşturacaktır.

```
name="acl"|version="2.3.1"|depends="attr"|acl
name="attr"|version="2.5.1"|depends=""|attr
name="audit"|version='3.1.1'|depends=""|audit
name="bash"|version="5.2.21"|depends="glibc,readline,ncurses"|bash
```

index Güncelleme

İndex güncelleme uzak(internet) depodaki paketlerin index listesinin yerelde tutulan index dosyasıyla eşitlemek işlemidir. Depoda olan paketlerin listesi yerelde tutulan index.lst dosyasındada olması gerekmektedir. Bu işlemi yapan klyupdate dosya içeriği aşağıdadır.

klyupdate

Debian sisteminde /etc/apt/source.list gibi bir yapı bulunmaktadır. Bu dosya içindeki depo adreslerine göre index oluşturmaktadır. Hazırladığımız paket sisteminde de /etc/kly/source.list dosyası kullanılmaktadır. Birden fazla index dosyasını birleştiren ve güncelleyen scriptimiz aşağıdadır.

```
#!/bin/sh
target="$1"
mkdir -p $target/etc/kly -p $target/tmp
rm -rf $target/etc/kly/index.lst
rm -rf $target/tmp/temp.lst
curl -Lo $target/etc/kly/index.lst \
https://github.com/kendilinuxunuyap/kly-binary-packages/releases/download/current/index.lst
```

klyupdate Kullanma

Script bir parametre almaktadır. Parametremiz --update veya -u olmalıdır. Bu scripti kullanarak /etc/kly/index.lst dosyasını github depomuzdaki paket listesiyle güncelleyecektir.

```
./klyupdate /home/user1/testiso
# /home/user1/testiso konumu hazırladığımız dağıtım konumudur.
# kendi siteminize uygun konum belirleyiniz.
```

Paket Kurma

Hazırlanan dağıtımda paketlerin kurulması için sırasıyla aşağıdaki işlem adımları yapılmalıdır.

1. Paketin indirilmesi
2. İndirilen paketin /tmp/kly/kur/ konumunda açılması
3. Açılan paket dosyalarının / konumuna yüklenmesi(kopyalanması)
    - Paketin bağımlı olduğu paketler varmı kontrol edilir
    - Yüklü olmayan bağımlılıklar yüklenir
4. Yüklenen paket bilgileri(name, version ve bağımlılık) **/var/lib/kly/index.lst** dosyasına eklenir.
5. Açılan paketin dosyalarının yüklendiğini konumları **/var/lib/kly/paket-version.lst** dosyasına eklenir.

Not: *3. adımdaki bağımlılık kontrolü ve yükjleme işlemleri tasarımımızda bulunmammaktadır. Fakat bu özellikler iyi bir paket sisteminde olmalıdır. Oluşturduğumuz* **klykur** *scripti sadece kurulum yapmaktadır.*

klykur Scripti

```
#!/bin/sh
#-------------------------------------------------------------------------------------------------------------
name="name=\"${1}\""
target=$2
mkdir -p $target
paket=$(echo $(cat $target/etc/kly/index.lst|grep $name)|cut -d"\"" -f2)
version=$(echo $(cat $target/etc/kly/index.lst|grep $name)|cut -d"\"" -f4)
depends=$(echo $(cat $target/etc/kly/index.lst|grep $name)|cut -d"\"" -f6)
# index dosyamızda paket aranıyor
if [ ! -n "${paket}" ]; then
echo "***********Paket Bulunamadı**********"; exit
fi

# 1. adım paketi indirme
mkdir -p $target/tmp/kly $target/tmp/kly/kur
rm -rf $target/tmp/kly/kur/*
curl -Lo $target/tmp/kly/kur/${paket}-${version}.tar.gz \
https://github.com/kendilinuxunuyap/kly-binary-packages/raw/master/${paket}/${paket}-${version}.kly
mkdir -p $target/var/lib/kly
cd $target/tmp/kly/kur/

# 2. adım paketi açma
tar -xf ${paket}-${version}.tar.gz
mkdir -p rootfs
tar -xf rootfs.tar.xz -C rootfs

# 3. adım paketi kurma
cp -prfv rootfs/* $target/

# 4. adım name version depends /var/lib/kly/index.lst eklenmesi
echo "name=\"${paket}\":"version=\"${version}\":"depends=\"${depends}\""">>$target/var/lib/kly/index.lst
# 5. adım paket içinde gelen paket dosyalarının dosya ve dizin yapısını tutan
# file index dosyanının /var/lib/kly/ konumuna kopyalanması
cp file.index $target/var/lib/kly/${paket}-${version}.lst
```

klykur Scriptini Kullanma

Script iki parametre almaktadır. İlk parametre paket adı. İkinci parametremiz ise nereye kuracağını belirten hedef olmalıdır. Bu scripti kullanarak readline paketi aşağıdaki gibi kurulabilir.

```
./klykur readline /home/user1/testiso
# /home/user1/testiso konumu hazırladığımız dağıtım konumudur.
# kendi siteminize uygun konum belirleyiniz.
```

Paket Kaldırma

Sistemde kurulu paketleri kaldırmak için işlem adımları şunlardır.

1. Paketin kullandığı bağımlılıkları başka paketler kullanıyor mu kontrol edilir. Eğer kullanılmıyorsa kaldırılır.
2. Paketin **/var/lib/kly/paket-verison.lst** dosyası içerisindeki dosyalar, dizinler kaldırılır.
3. Kaldırılan dosyalardan sonra **/var/lib/kly/paket-verison.lst** dosyası silinir.
4. sistemde kurulu paketleri tutan **/var/lib/kly/index.lst** dosyasından ilgili paket satırı kaldırılmalıdır.

Not: Paket kaldırma işlemlerini yapan **klykaldir** scriptinde 1. adımdaki bağımlılık kontrolü yapma ve kaldırma işlemi yapılmamaktadır. Fakat iyi bir paket sisteminde mutlaka yapılmalıdır. Çünkü sistemde ihtiyaç olmayan paketler kalacaktır.

klykaldir scripti

```
#-----------------------------------------------------------------------------------------------------------------
#!/bin/sh
name="name=\"${1}\""
target=$2
mkdir -p $target
paket=$(echo $(cat $target/etc/kly/index.lst|grep $name)|cut -d"\"" -f2)
version=$(echo $(cat $target/etc/kly/index.lst|grep $name)|cut -d"\"" -f4)
depends=$(echo $(cat $target/etc/kly/index.lst|grep $name)|cut -d"\"" -f6)
# index dosyamızda paket aranıyor
if [ ! -n "${paket}" ]; then
        echo "***********Paket Bulunamadı**********"; exit
fi
# Bağımlılıkları başka paketler kullanıyor mu kontrol edilir
# Başka paketler kullanılıyorsa silinmemeli. Bu işlemin kodları yazılmadı.
echo "${paket}-${version} bağımlılık kontrolü yapılacak"

# 2. adım Paketin paket-version.lst dosyası içerisindeki dosyalar kaldırılır.
if [ -f "$target/var/lib/kly/${paket}-${version}.lst" ]; then
        cat $target/var/lib/kly/${paket}-${version}.lst | while read dosya ;
        do
                if [[ -f "$target/$dosya" ]] ; then rm -f "$target/$dosya"; fi
        done
fi
# 3. adım /var/lib/kly/paket-version.lst dosyası silinir.
rm -f $target/var/lib/kly/${paket}-${version}.lst

# 4. adım /var/lib/kly/index.lst dosyasından ilgili paket satırı kaldırılır.
sed -i "/name=\"${paket}\"/d" $target/var/lib/kly/index.lst

echo "********** ${paket}-${version}  Paketi Kaldırıldı **********"
```

klykaldir Kullanma

```
./klykaldir readline /home/user1/testiso
# /home/user1/testiso konumu hazırladığımız dağıtım konumudur.
# kendi siteminize uygun konum belirleyiniz.
```

Yardımcı Konular

apt Paket Sistemi

apt paket sistemi debian tabanlı sistemlerde paket yönetimi için kullanılan bir uygulamadır. Temel işlemler şunlardır.

Yerel Paket İndexini Güncelleme

```
sudo apt  update
```

Paket Yükleme

```
sudo apt install paket_adi
```

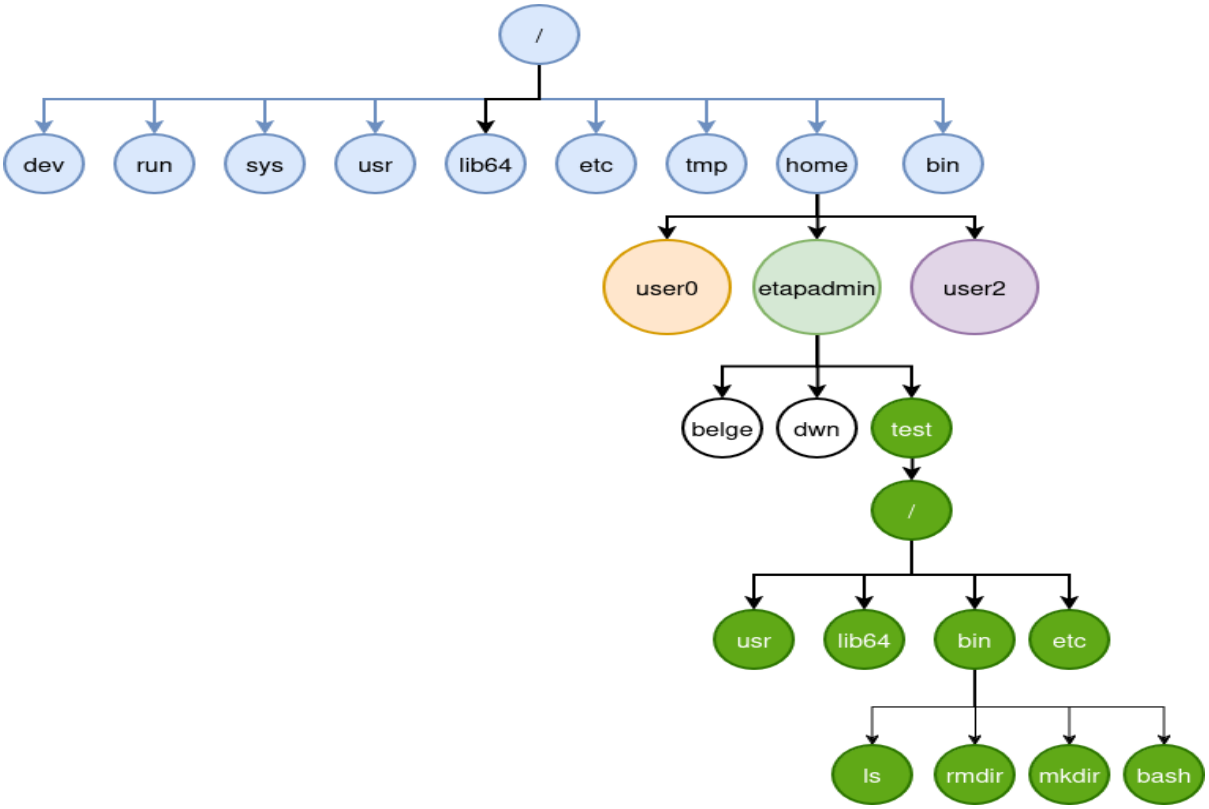
Paket Silme

```
sudo apt remove paket_adi
```

Chroot Nedir?

chroot komutu çalışan sistem üzerinde belirli bir klasöre root yetkisi verip sadece o klasörü sanki linux sistemi gibi çalıştıran bir komuttur. Sağladığı avantajlar çok fazladır. Bunlar;

- Sistem tasarlama
- Sitem üzerinde yeni dağıtımlara müdahale etme ve sorun çözme
- Kullanıcı kendine özel geliştirme ortamı oluşturabilir.
- Yazılım bağımlıkları sorunlarına çözüm olabilir.
- Kullanıcıya sadece kendisine verilen alanda sınırsız yetki verme vb.



Yukarıdaki resimde user1 altında wrk dizini altına yeni bir sistem kurulmuş gibi yapılandırmayı gerçekleştirmiş.

/home/etapadmin/test dizinindeki sistem üzerinde sisteme erişmek için;

```
# sisteme erişim yapıldı.
sudo chroot /home/etapadmin/test
```

/home/etapadmin/test dizinindeki sistem üzerinde sistemi silmek için;

```
# sistem silindi
sudo rm -rf /home/etapadmin/test
```

Yeni sistem tasarlamak ve erişmek için temel komutları ve komut yorumlayıcının olması gerekmektedir. Bunun için bize gerekli olan komutları bu yapının içine koymamız gerekmektedir. Örneğin ls komutu için doğrudan çalışıp çalışmadığını ldd komutu ile kontrol edelim.

Görüldüğü gibi ls komutunun çalışması için bağımlı olduğu kütüphane dosyaları bulunmaktadır. Bağımlı olduğu dosyaları yeni oluşturduğumuz sistem dizinine aynı dizin yapısında kopyalamamız gerekmektedir. Bu dosyalar eksiksiz olursa ls komutu çalışacaktır. Fakat bu işlemi tek tek yapmamız çok zahmetli bir işlemdir. Bu işi yapacak script dosyası aşağıda verilmiştir.

Bağımlılık Scripti

lddscript.sh

```
#!/bin/bash

# Betik iki parametre bekler: kopyalanacak dosya ve hedef klasör
if [ ${#} != 2 ]; then
        echo "Kullanım: $0 PATH_TO_BINARY hedef_klasor"
        exit 1
fi

path_to_binary="$1"
target_folder="$2"

# Dosya yoksa işlem durur
if [ ! -f "${path_to_binary}" ]; then
        echo "Dosya '${path_to_binary}' bulunamadı. İşlem iptal ediliyor!"
        exit 1
fi

# Dosyayı kopyala
echo "Dosya kopyalanıyor..."
cp --parents -v "${path_to_binary}" "${target_folder}"

# Bağımlı kütüphaneleri kopyala
echo "Kütüphaneler kopyalanıyor..."
ldd "${path_to_binary}" | awk -F'[> ]' '{print $(NF-1)}' | while read -r lib; do
        [ -f "$lib" ] && cp -v --parents "$lib" "${target_folder}"
done
```

Basit Sistem Oluşturma

Bu örnekte kullanıcının(etapadmin) ev dizinine(/home/etapadmin) test dizini oluşturuldu ve işlemler yapıldı. ls, rmdir, mkdir ve bash komutlarından oluşan sistem hazırlama.

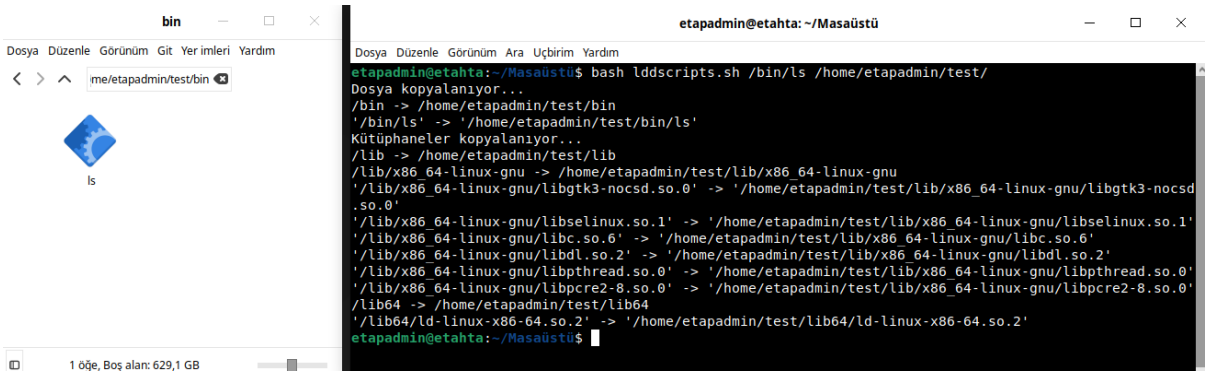
Sistem Dizinin Oluşturulması

```
# ev dizinine test dizini oluşturuldu.
mkdir /home/etapadmin/test/
```

/home/etapadmin/ dizinine **Bağımlılık Scripti** kodunu **lddscripts.sh** oluşturalım.

ls Komutu

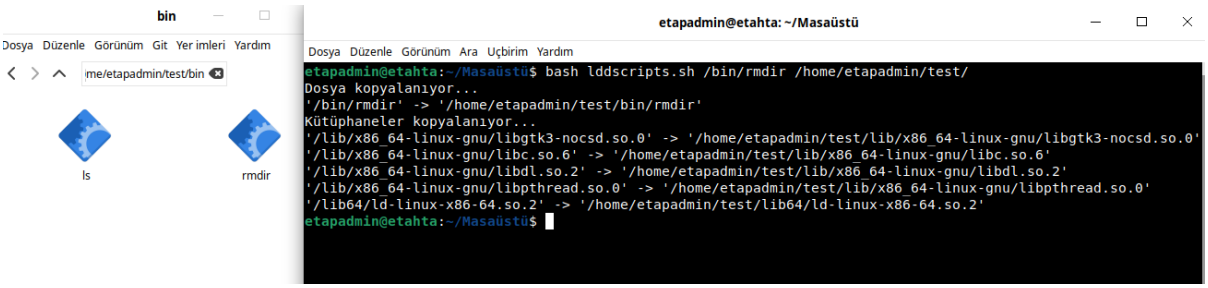
```
bash lddscripts.sh /bin/ls /home/etapadmin/test/
```



Bu işlemi diğer komutlar içinde sırasıyla yapmamız gerekmektedir.

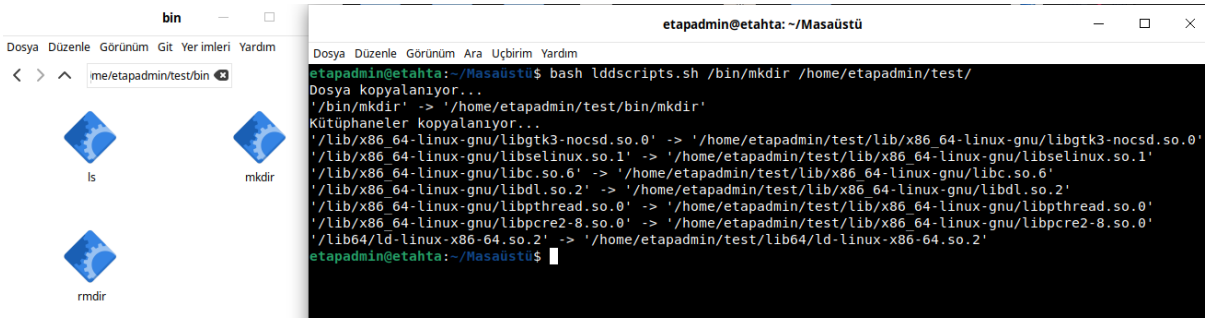
rmdir Komutu

```
bash lddscripts.sh /bin/rmdir /home/etapadmin/test/
```



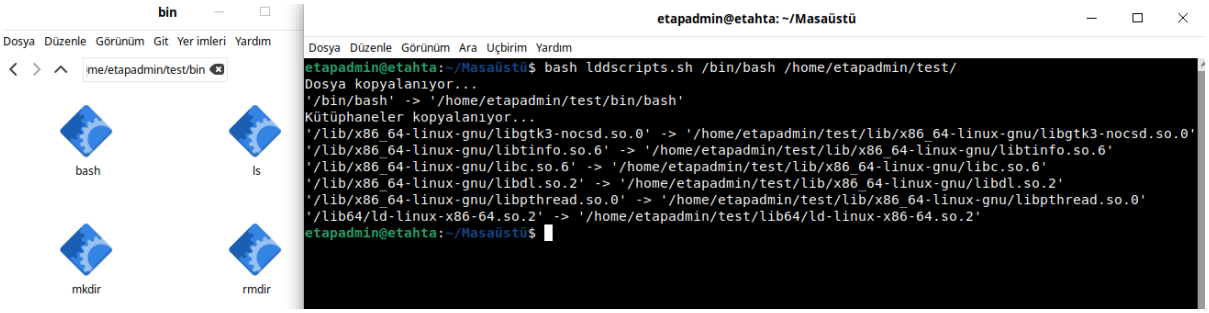
mkdir Komutu

```
bash lddscripts.sh /bin/mkdir /home/etapadmin/test/
```



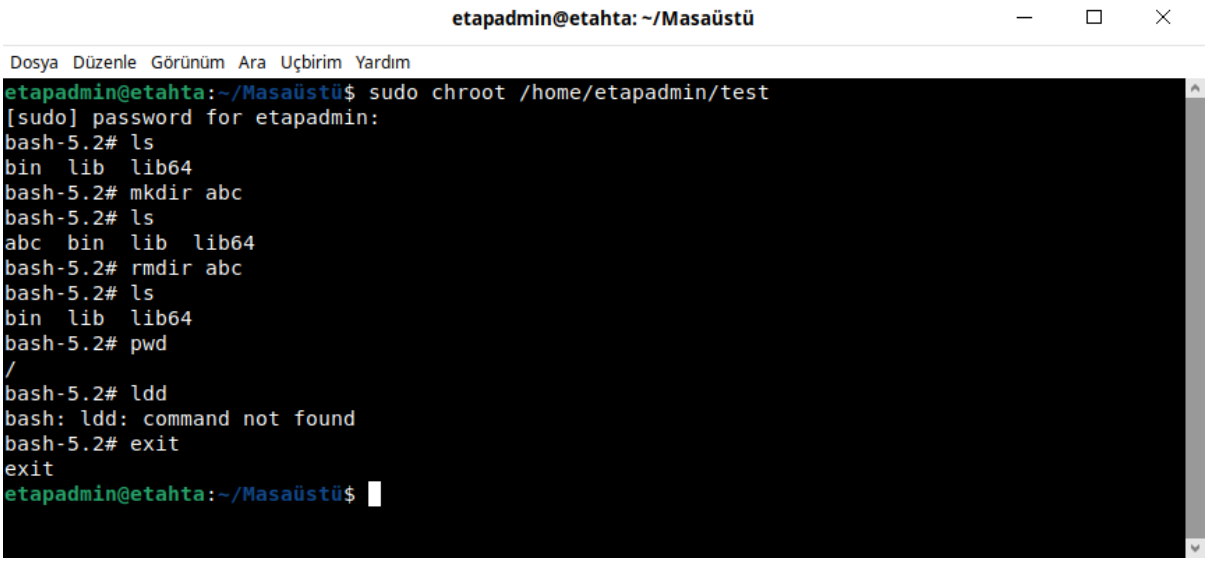
bash Komutu

```
# bash komutu ve bağımlılığı kopyalandı.
bash lddscripts.sh /bin/bash /home/etapadmin/test/
```



chroot Sistemde Çalışma

```
sudo chroot /home/etapadmin/test komutunu kullanmalıyız.
```



- **abc** dizini oluşturuldu, **abc** dizini silindi, **pwd** komutuyla konum öğrenildi, **ldd** komutu sistemimizde olmadığından hata verdi.
- Çıkış için ise **exit** komutu kullanılarak sistemden çıkıldı.

Kaynak:
https://stackoverflow.com/questions/64838052/how-to-delete-n-characters-appended-to-ldd-list
https://app.diagrams.net/

Kaynak Kod Derleme

Bir uygulamanın kodları genellikle çalışmaz(python benzeri kodlar istisna). Bu kodlardan sistemlerin çalışması için çalışabilir dosyalar üretilir(linuxta ikili dosya, elf, windowsta exe, com vb.). Bu çalışabilir dosyaları koddan oluştururken iki faklı şekilde oluşturabiliriz.

1. **Paylaşımlı Derleme(dynamic):** Kendine lazım olan kütüphaneleri sistem üzerindeki başka uygulamalarla ortak kullanır.
2. **Paylaşımsız, gömülü(static):** Kendisine lazım olan kütütphaneleri kendi içinde barındırır(porable uygulama gibi).

Şimdi aşağıdaki kaynak kodumuzu iki farklı yöntemle derleyelim.

```
//main.c dosyamız
#include <stdio.h>
void main(){
    printf("Merhaba\n");
}
```

2-Paylaşımlı Derleme(dynamic):

Derlenen uygulama sistemde bulunan kütüphaneleri kullanacak şeklide derlenmesidir. Uygulama boyutu küçüktür, taşınabirliği sınırlanabilir. Aşağıdaki gibi derlenir.

```
gcc -o main main.c
```

main.c kodumuzu **main** adında ikili çalışabilir dosyaya dönüşütürdük. **ldd** komutuyla **main** dosyasının kullandığı kütüphaneler öğrenilir.

```
unset LD_PRELOAD
ldd ./main
        linux-vdso.so.1 (0x00007ffdb3bb9000)
        libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007f2c53fe0000)
        /lib64/ld-linux-x86-64.so.2 (0x00007f2c541e7000)
```

Burada **libc.so.6** ve **ld-linux-x86_64.so.2** dosyaları **glibc** tarafından sağlanır. Derlenmiş dosyanın çalışması için tüm bağımlılıklarının sistemde bulunması gereklidir.Sadece gerekli olan kütüphaneleri görmek için **readelf -d** komutu kullanılabilir. Aşağıda gerekli olan kütüphaneler listeleniyor.

```
readelf -d ./main | grep -i needed
       0x0000000000000001 (NEEDED)             Paylaşımlı kitaplık: [libc.so.6]
```

ldconfig

Sistemdeki kütüpkhanelerin konumlarını **/etc/ld.so.conf** dosyasına bakarak belirler.

```
#/etc/ld.so.conf dosyası
/usr/lib64
/usr/lib
/lib64
/lib
```

Kütüphanelerde değişiklik yapılmışsa ve hemen bu değişikliği sistemin görmesini istersek **ldconfig** komutu kullanılmalıdır.

2-Paylaşımsız, gömülü(static):

Derlenen uygulama sistemde bulunan ve çalışması için gerekli olan kütüphaneleri uygulama içine dahil eden bir derleme yöntemidir. Uygulamamızı static derlemek için **-static** parametresi ekleyerek derlenir.

```
gcc -o main main.c -static
```

Paylaşımlı(dynamic) derleme işleminde bağımlı olduğu dosyaları **ldd** komutunu kullanarak öğrenmiştik. Şimdi paylaşımsız derlediğimiz **main** dosyasında bağımğımlı olduğu kütüphaneleri kontrol ediyoruz.

```
ldd main
    not a dynamic executable
```

Bağımlı kütüphaneler yerine **not a dynamic executable** mesajı gördük. Bunun anlamı çalışması için hiçbir kütüphaneye ihtiyaç duymaz. Bu bir avantaj ve taşınabirliği artırır. Devevantajı ise boyutu büyük olur. İhtiyaca göre paylaşımlı veya paylaşımsız derleme tercih edilir.

Derleme Araçları

Linux sistemlerinde kodlarımızı derlemek kullanılan uygulamalara derleme araçları denir. Birden fazla araç olsada en temel derleme araçları **make**, **cmake**, **meson**, **python**.

1-make

make, yazılım geliştirme süreçlerinde sıkça kullanılan bir araçtır. Özellikle C ve C++ gibi dillerde projelerin derlenmesi ve yönetilmesi için kullanılır.

Aşağıdaki C kodumuzu(merhaba.c dosyası) make ile nasıl derleyeceğimizi anlatalım;

```
#include <stdio.h>
int main(){
    puts("Merhaba");
    return 0;
}
```

Makefile Nedir?

Makefile, make aracının nasıl çalışacağını belirten bir dosyadır. İçinde hedefler, bağımlılıklar ve komutlar bulunur. Örneğin, bir C programını derlemek için aşağıdaki gibi basit bir Makefile oluşturabilirsiniz. Makefile ve merhaba.c dosyası aynı konumda olmalıdır.

```
CC=gcc
CFLAGS=-I.
all: program
program: merhaba.o
        $(CC) -o program merhaba.o
merhaba.o: merhaba.c
        $(CC) -c merhaba.c $(CFLAGS)
clean:
        rm -f *.o program
```

merhaba dosyasından **program** adında bir ikili dosya oluşturmak için aşağıdaki komutlar Makefile ve merhaba.c dosyasının olduğu konumda çalıştırılır.

```
make
make install
```

Genellikle Makefile dosyası olmaz. Onun yerine **configure.ac** dosyası olur. **configure.ac** dosyasından **Makefile** dosyası elde etmek için, öncelikle **autoconf** ve **automake** araçlarının sisteminizde kurulu olması gerekmektedir.

autoreconf -fvi komutu çalıştırılıarak configure dosyamızı üretir.Bu araçlar, yapılandırma dosyalarınızı işleyerek gerekli **Makefile** dosyalarını oluşturmanıza yardımcı olur.

configure komutunun devamında **--prefix** dışında başka parametreleride olabilir. Bu paramatreleri "Read.me" dosyası içerisinde bulunabilir veya **configure --help** komutu kaynak kodların olduğu konumda kullanılarak görülebilir. Bu kaynak kod aşağıdaki gibi derlenir.

```
$ autoreconf -fvi
$ ./configure --prefix=/usr
$ make
$ make install
```

2-cmake

CMake, projelerin derlenmesi ve yapılandırılması için kullanılan bir araçtır. Bir paketi CMake ile derlemek için genellikle aşağıdaki adımları izleriz:

İlk olarak, projenin kök dizininde bir CMakeLists.txt dosyası oluşturun. Ardından, CMake komutunu kullanarak projeyi yapılandırın ve derleyin. Örneğin:

```
    mkdir compile_packages
    cd compile_packages
    cmake ..
    make

Bu adımları takip ederek, CMake ile paketinizi başarıyla derleyebilirsiniz.
```

CMake'in esnekliği ve kullanım kolaylığı sayesinde paketlerin derlenmesi ve yapılandırılması oldukça kolay hale gelir.

Bu kaynak kod aşağıdaki gibi derlenir:

```
mkdir compile_packages
cd compile_packages
cmake ..
make
make install
```

3-meson

Meson, modern bir yapılandırma betik dili ve Ninja ise hızlı bir derleyici aracıdır. Bir paketi derlemek için öncelikle Meson ile yapılandırma dosyalarını oluşturmanız gerekir. Ardından, Ninja derleyici aracını kullanarak bu yapılandırmayı derleyebilirsiniz.

İlk olarak, projenizin dizinine gidin ve Meson ile yapılandırma dosyalarını oluşturun:

```
meson setup builddir --prefix=/usr
```

Sonra, oluşturulan builddir dizinine geçin ve Ninja ile derlemeyi başlatın:

```
ninja -C builddir
```

Bu adımları takip ederek Meson ve Ninja kullanarak paketinizi başarılı bir şekilde derleyebilirsiniz.

Bu kaynak kod aşağıdaki gibi derlenir:

```
# paketin yükleneceği konum
INSTALL_ROOT=/
meson setup builddir --prefix=/usr
ninja -C builddir
INSTALL_ROOT=$INSTALL_ROOT ninja -C builddir install
```

4-python

Python ile paket derlemek için genellikle **setuptools** ve **distutils** gibi kütüphaneler kullanılır. Öncelikle, projenizin kök dizininde bir **setup.py** dosyası oluşturmanız gerekir. Bu dosya, paketin yapılandırmasını ve gereksinimlerini tanımlar.

Örnek bir setup.py dosyası aşağıdaki gibi olabilir:

```
from setuptools import setup

setup(
    name='paket_adi',
    version='1.0',
    packages=['paket'],
    install_requires=[
        'bağımlılık_paketi',
    ],
)
```

Ardından, terminalde projenizin bulunduğu dizine giderek aşağıdaki komutu çalıştırarak paketi derleyebilirsiniz:

Bu komut, paketi dist klasörüne derleyecektir. Artık paketiniz hazır ve dağıtıma uygun hale gelmiştir.

BusyBox

BusyBox, https://busybox.net/about.html adresteki bilgilere göre birçok temel Unix aracını tek bir ikili dosya içinde sunan, küçük ve esnek bir programdır. Çoğunlukla **initramfs** sistemlerinde ve gömülü Linux dağıtımlarında tercih edilir. BusyBox ile komutlar şu şekilde çalıştırılabilir:

```
busybox [komut]
```

Bir komutu doğrudan çalıştırabilmek için BusyBox'u o komut adıyla sembolik bağlamak mümkündür. Örneğin, **tar** komutunu BusyBox üzerinden çalıştırmak için:

```
ln -s /bin/busybox ./tar
./tar
```

Burada busybox içindeki gömülü olan tar kullanmayı gördük. Aslında aklımaza gelen her komutun busybox içinde gömülü hali vardır. Ama bu tüm komutlar için ve tüm parametreleri için geçerli değildir. Busybox içindeki tüm komutların kısayolunu(sembolik bağı) eklemek için aşağıdaki komut kullanılır.

```
busybox --install -s /bin
```

Derleme

Gömülü sistem tasarımı veya **initramfs** içinde kullanılacak busybox **paylaşımsız(static)** derlenir. Bu şekilde derlendiğinde glibc kütüphanelerine ihtiyaç duymadan çalışacaktır.

Şimdi busybox derlemek için https://busybox.net/ adresinden **stable** başlığı altındaki kaynak kodlarını indiriniz.

```
# kaynak kod indirilir
wget https://busybox.net/downloads/busybox-1.36.1.tar.bz2

# indirilen dosya açılır
tar -xvjf busybox-1.36.1.tar.bz2

# açılan kaynak dosyalarının bulunduğu dizine geçilir
cd busybox-1.36.1/
```

Şimdi kodlarımızı indirdik. derlemek için aşağıdaki komutları çalıştıralım.

```
# varsayılan yapılandırmayı uygula
make defconfig

# static derleme yapacaksak  sed ile .config dosyasını düzenliyoruz.
sed -i "s|.*CONFIG_STATIC_LIBGCC .*|CONFIG_STATIC_LIBGCC=y|" .config
sed -i "s|.*CONFIG_STATIC .*|CONFIG_STATIC=y|" .config

# derliyoruz
make

# Static derleme sonucu aşağıdaki gibi görünür.
ldd /bin/busybox
özdevimli bir çalıştırılabilir değil
```

Derleme tamamlandığında, oluşturulan **busybox** ikili dosyası kaynak dizininde bulunur.

OpenRC

Openrc sistem açılışında çalışacak uygulamaları çalıştıran servis yöneticisidir.

Çalıştırılması

Openrc servis yönetiminin çalışması için boot parametrelerine yazılması gerekmektedir. **/boot/grub.cfg** içindeki **linux /vmlinuz init=/usr/sbin/openrc-init root=/dev/sdax** olan satırda **init=/usr/sbin/openrc-init** yazılması gerekmektedir. Artık sistem openrc servis yöneticisi tarafından uygulamalar çalıştırılacak ve sistem hazır hale getirilecek.

openrc Kullanımı

Servis etkinleştirip devre dışı hale getirmek için **rc-update** komutu kullanılır. Aşağıda **udhcpc** internet servisi örnek olarak gösterilmiştir. **/etc/init.d/** konumunda **udhcpc** dosyamızın olması gerekmektedir.

```
# servis etkinleştirmek için
rc-update add udhcpc boot
# servisi devre dışı yapmak için
rc-update del udhcpc boot
# Burada udhcpc servis adı, boot ise runlevel adıdır.
```

Elle **/etc/runlevels/** altında bulunan **boot**, **default**, **nonetwork**, **shutdown**, **sysinit** dizinlerine **/etc/init.d/** dizininin altındaki dosyaları **ln** komutuyla kısayol yaparsak **rc-update add** komutunun yaptığı görevi yapmış oluruz. Kısayolu silersek **rc-update del** komutunun görevini yapmış oluruz.

/etc/runlevels/ altında bulunan **boot**, **default**, **nonetwork**, **shutdown**, **sysinit** dizinler servis dosyalarımızın hangi sırayla çalışacağını belirleyen dizinlerdir. Mesela **boot** dizini ilk açılış sırasında çalışacak olan servis dosyalarının konulacağı dizindir.

Servisleri başlatıp durdurmak için ise **rc-service** komutu kullanılır.

```
rc-service udhcpc start
# veya şu şekilde de çalıştırılabilir.
/etc/init.d/udhcpc start
```

Servislerin durumunu öğrenmek için **rc-status** komutu kullanılır. Ayrıca sistemdeki servislerin sonraki açılışta hangisinin başlatılacağını öğrenmek için parametresiz olarak **rc-update** kullanabilirsiniz.

```
# şu an hangi servislerin çalıştığını gösterir
rc-status
# sonraki açılışta hangi servislerin çalışacağını gösterir
rc-update
```

Sistemi kapatmak veya yeniden başlatmak için **openrc-shutdown** komutunu kullanabilirsiniz.

```
openrc-shutdown -p 0     # kapatmak için
openrc-shutdown -r 0     # yeniden başlatmak için
```

Servis Dosyası

Openrc servis dosyaları basit birer **bash** betiğidir. Bu betikler **openrc-run** komutu ile çalıştırılır ve çeşitli fonksiyonlardan oluşabilir. Servis dosyaları **/etc/init.d** içerisinde bulunur. Servisleri ayarlamak için ise **/etc/conf.d** içerisine aynı isimle ayar dosyası oluşturabiliriz.

Çalıştırılacak komut, komut parametreleri ve **pidfile** dosyamızı aşağıdaki gibi belirtebiliriz.

```
description="Ornek servis"
name="ornek-servis"
command=/usr/bin/ornek-servis
command_args=--parametre
pidfile=/run/ornek-servis.pid
command_background=true
start_stop_daemon_args="--quiet"
output_log="/var/log/ornek-servis.log"
error_log="/var/log/ornek-servis.err"
```

Eğer bu scriptin sadece boot sırasında bir kere çalışması yetiyorsa, o zaman **command_background=true** ayarını kaldırın. OpenRC aslında "oneshot" türü servisleri destekler. Böyle bir senaryoda **pidfile=/run/ornek-servis.pid** gerek yok. Mesajların gözükmemesi için **start_stop_daemon_args="--quiet"** kullanılır.

Bununla birlikte **start**, **stop**, **status**, **reload**, **start_pre**, **stop_pre** gibi fonksiyonlar da yazabiliriz.

```
start(){
    ebegin "Starting ${RC_SVCNAME}"
    start-stop-daemon --start --pidfile "/run/servis.pid" --exec /usr/bin/ornek-servis --parametre
}
```

Servis bağımlılıklarını belirtmek için ise **depend** fonksiyonu kullanılır.

```
depend() {
  need localmount
  after dbus
}
```

Qemu Kullanımı

qemu açık kaynaklı Virtual Box, Vmware benzeri sanallaştırma aracıdır.

Sisteme Kurulum

```
sudo apt update
sudo apt install qemu-system-x86  qemu-utils
```

qemu Kullanımı

```
# 30GB disk oluşturuldu.
qemu-img create disk.img 30G
qemu-system-x86_64 --enable-kvm -hda disk.img -m 2G -cdrom etahta.iso
# qemu-system-x86_64 --enable-kvm -hda disk.img -m 2G #sadece disk ile çalıştırılıyor
# qemu-system-x86_64 -m 2G -cdrom etahta.iso          #sadece iso dosyası ile çalıştırma
```

Sistem Hızlandırılması

--enable-kvm eğer sistem disk ile çalıştırıldığında bu parametre eklenmezse yavaş çalışacaktır.

Boot Menu Açma

Sistemin diskten mi imajdan mı başlayacağını başlangıçta belirlemek için boot menu gelmesini istersek aşağıdaki gibi komut satırına seçenek eklemeliyiz.

```
qemu-system-x86_64 --enable-kvm -cdrom distro.iso -hda disk.img -m 4G -boot menu=on
```

Uefi kurulum için:

sudo apt-get install ovmf

qemu-system-x86_64 --enable-kvm -bios /usr/share/ovmf/OVMF.fd -cdrom distro.iso -hda disk.img -m 4G -boot menu=on

qemu Host Erişimi:

İstemci bilgisayar ip'si:10.0.2.15 ve ana bilgisayar 10.0.0.2 olarak ayarlıyor.

vmlinuz ve initrd

qemu ile vmlinuz ve initrd.img dosyaları iso olmadan test edilebilir.

```
qemu-system-x86_64 --enable-kvm -kernel /boot/vmlinuz-5.17 -initrd /home/deneme/initrd.img -append "quiet" -m 512m
```

qemu Terminal Yönlendirmesi

```
qemu-system-x86_64 --enable-kvm -kernel vmlinuz -initrd initrd.img -m 3G -serial stdio -append "console=ttyS0"
```

Diskteki Sistemin Açılışını Terminale Yönlendirme

```
qemu-system-x86_64 -nographic -kernel boot/vmlinuz -hda disk.img -append console=ttyS0
```

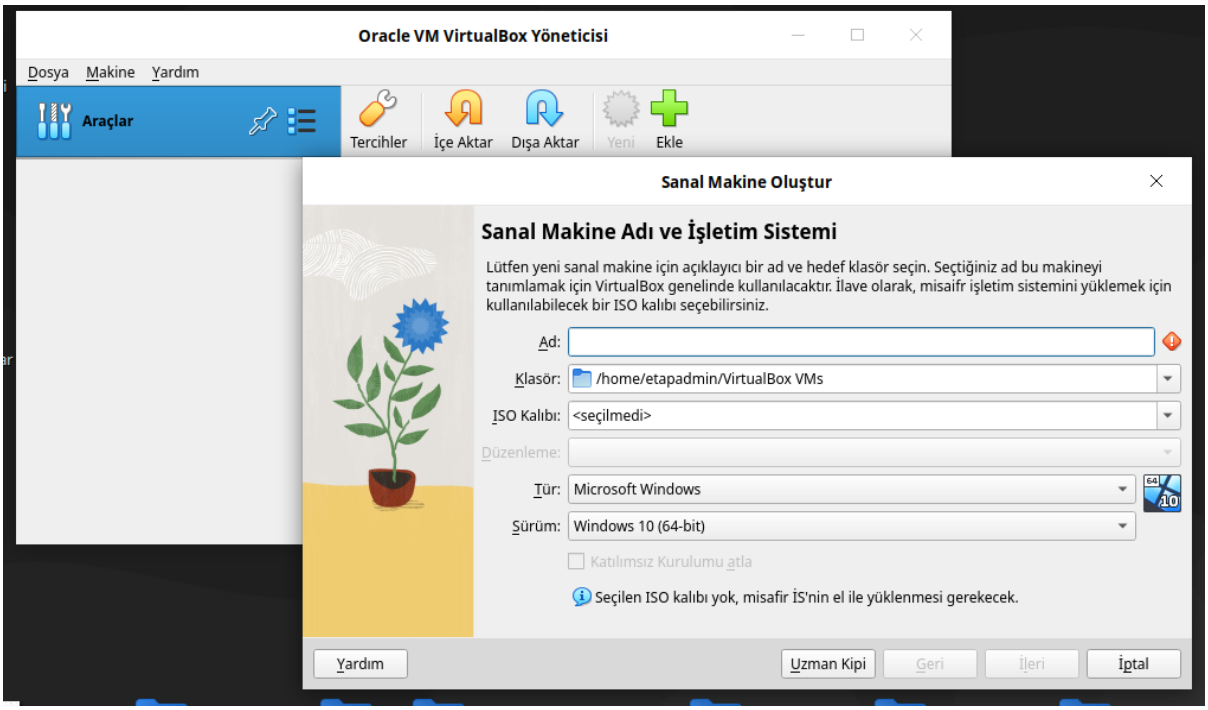
Kaynak: | https://www.ubuntubuzz.com/2021/04/how-to-boot-uefi-on-qemu.html

VirtualBox

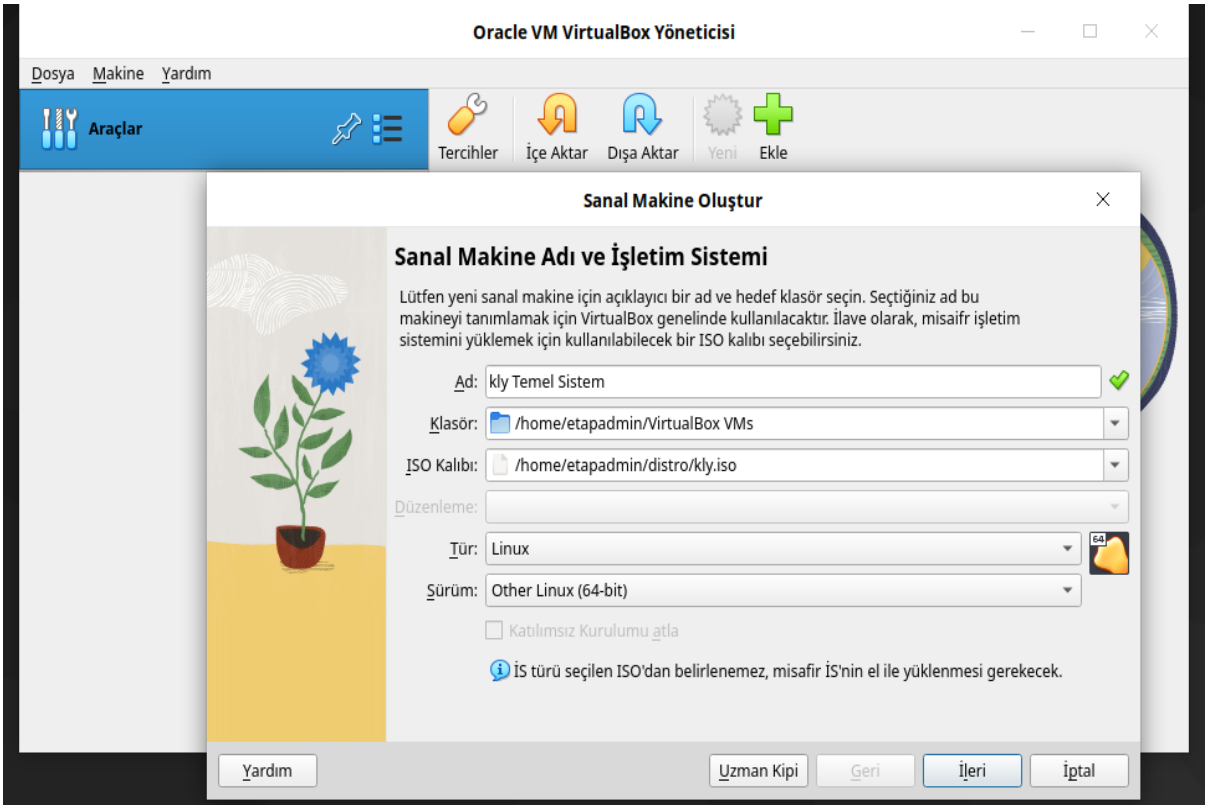
VirtualBox, Oracle tarafından geliştirilen bir açık kaynaklı sanallaştırma yazılımıdır. Bilgisayarınızda çalışan bir işletim sisteminin (örneğin Windows, Linux, macOS) içinde başka bir işletim sistemi çalıştırmanıza izin verir. Bu çalıştırdığınız ikinci işletim sistemi (konuk/guest) kendi sanal donanımına sahipmiş gibi davranır.

```
sudo apt update                          # index güncellenir
sudo apt install virtualbox-7.0          # Sisteme virtualbox kurulur
```

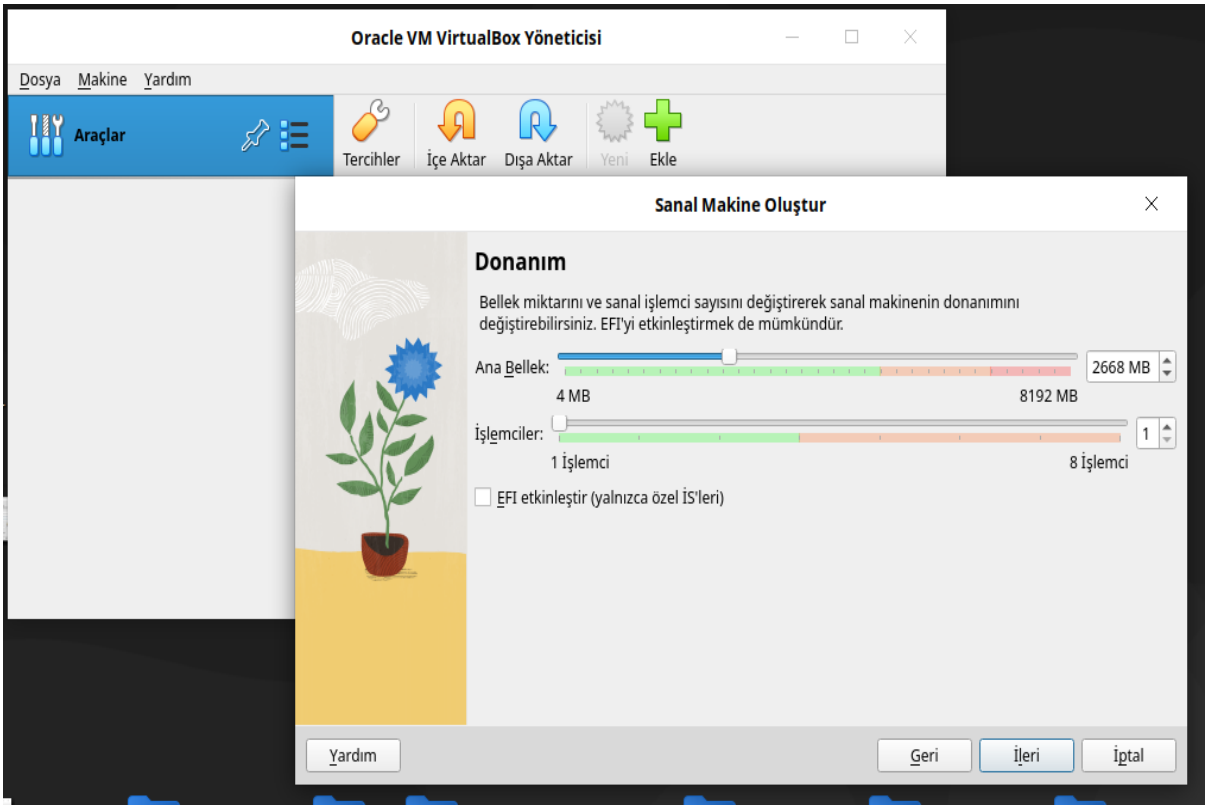
Ekle seçeneğini seçin. Aşağıdaki gibi bir ekran gelecektir.



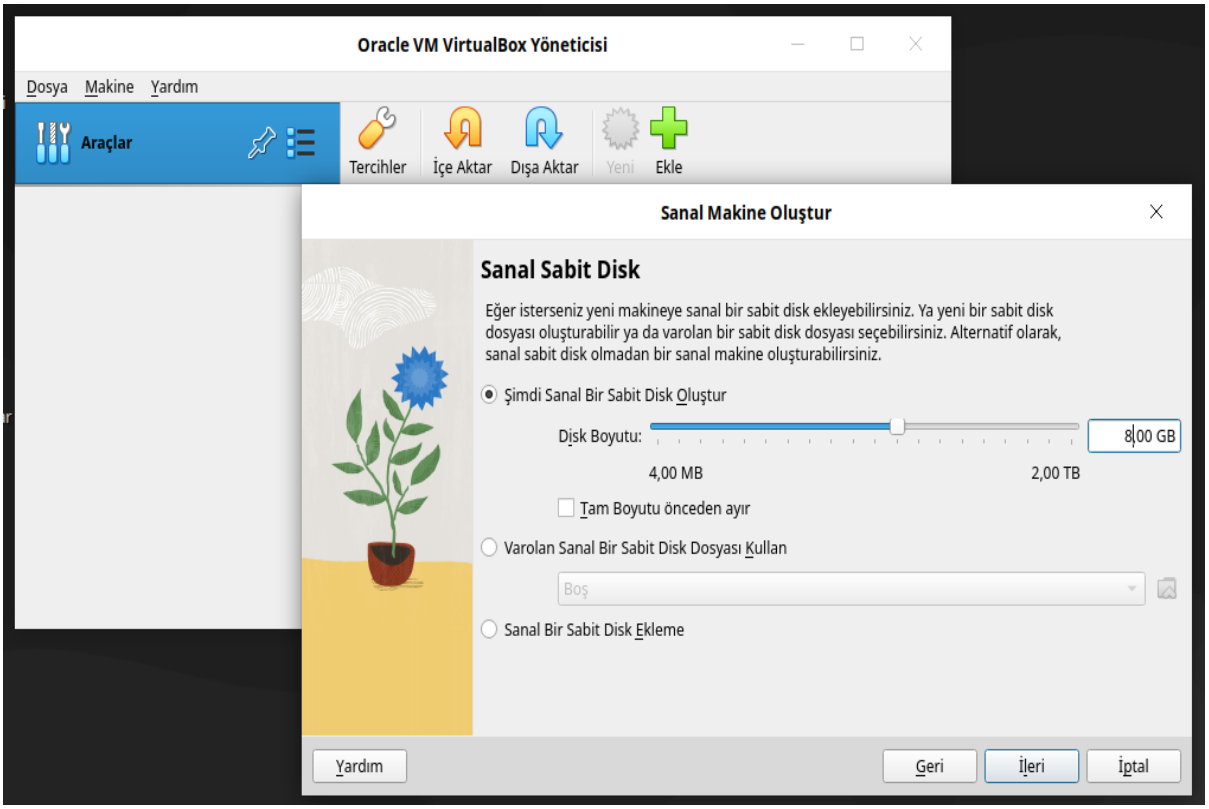
Aşağıdaki gibi seçenekleri doldurunuz.



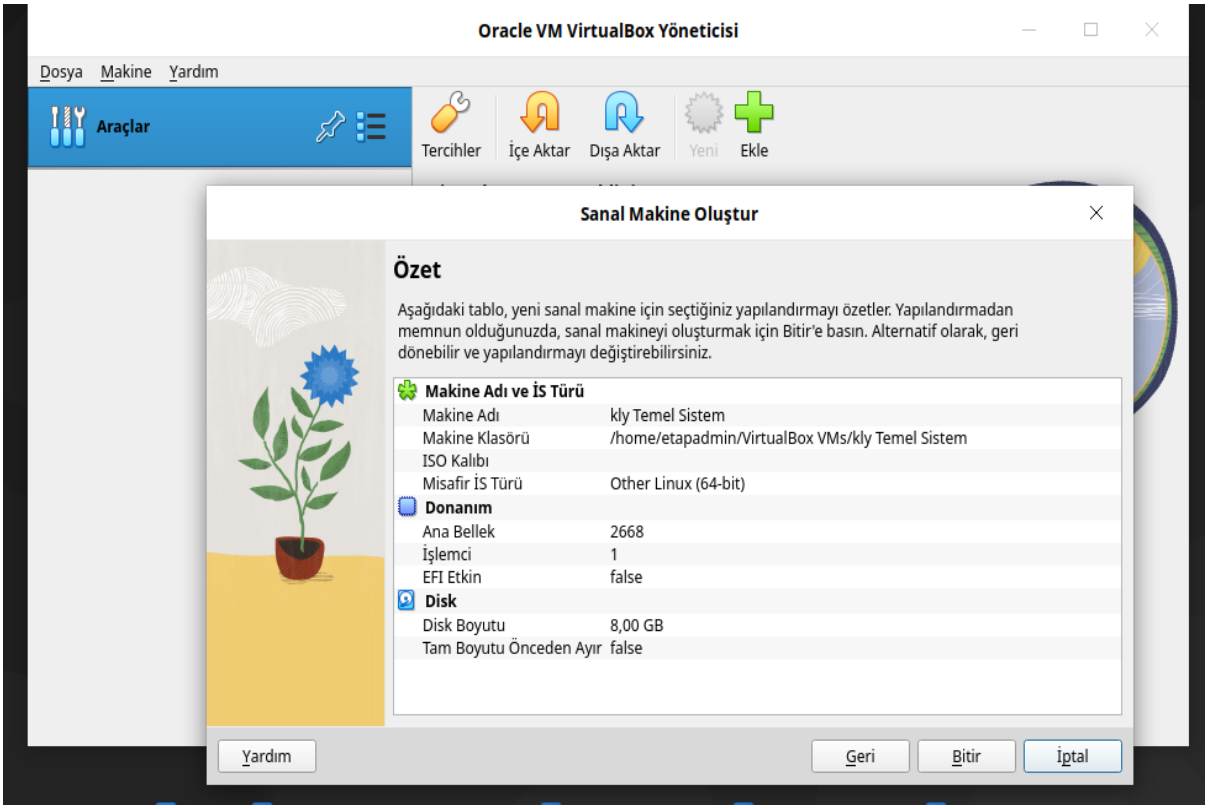
İleri seçeneği seçilir ve aşağıdaki gibi ekrana gelirsiniz. Bellek miktarını aşağıdaki gibi belirleyiniz. **İleri** seçeneği seçilir.



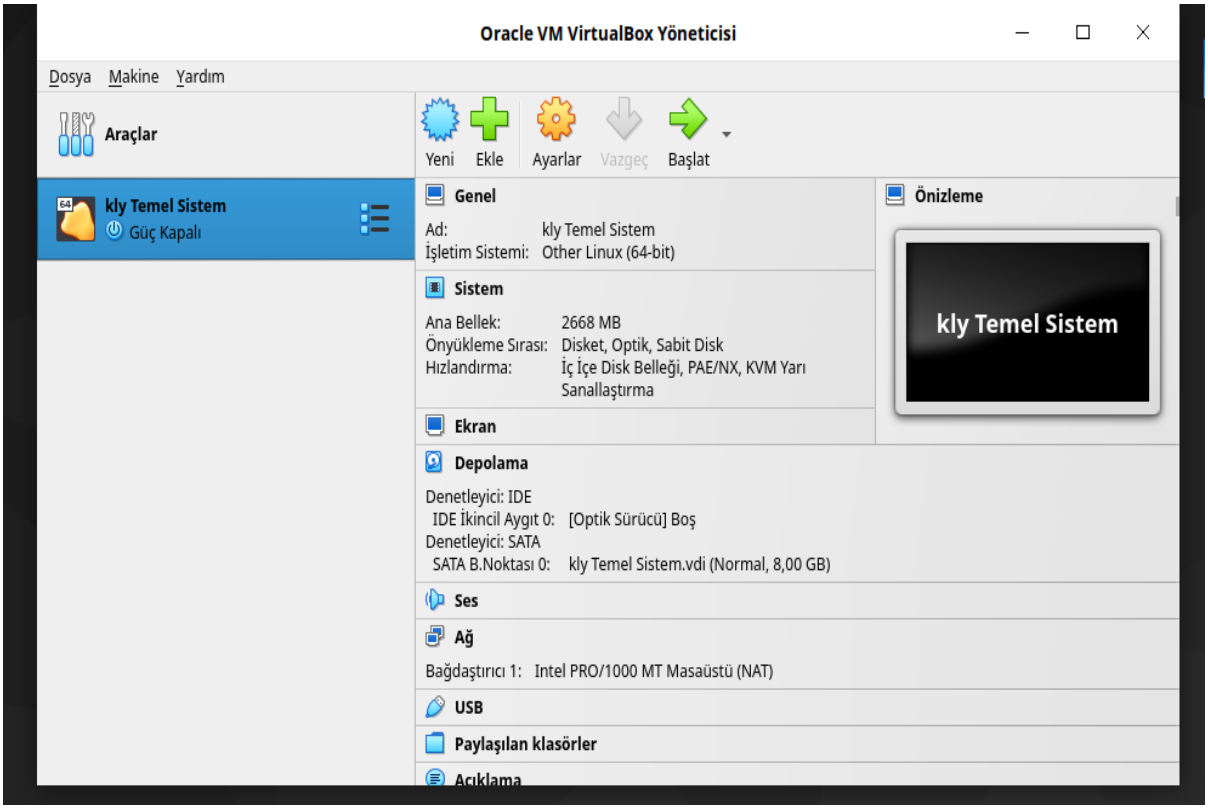
Disk boyutu belirlenir. 8GB bu sistem için yeterli. **İleri** seçeneği seçilir.



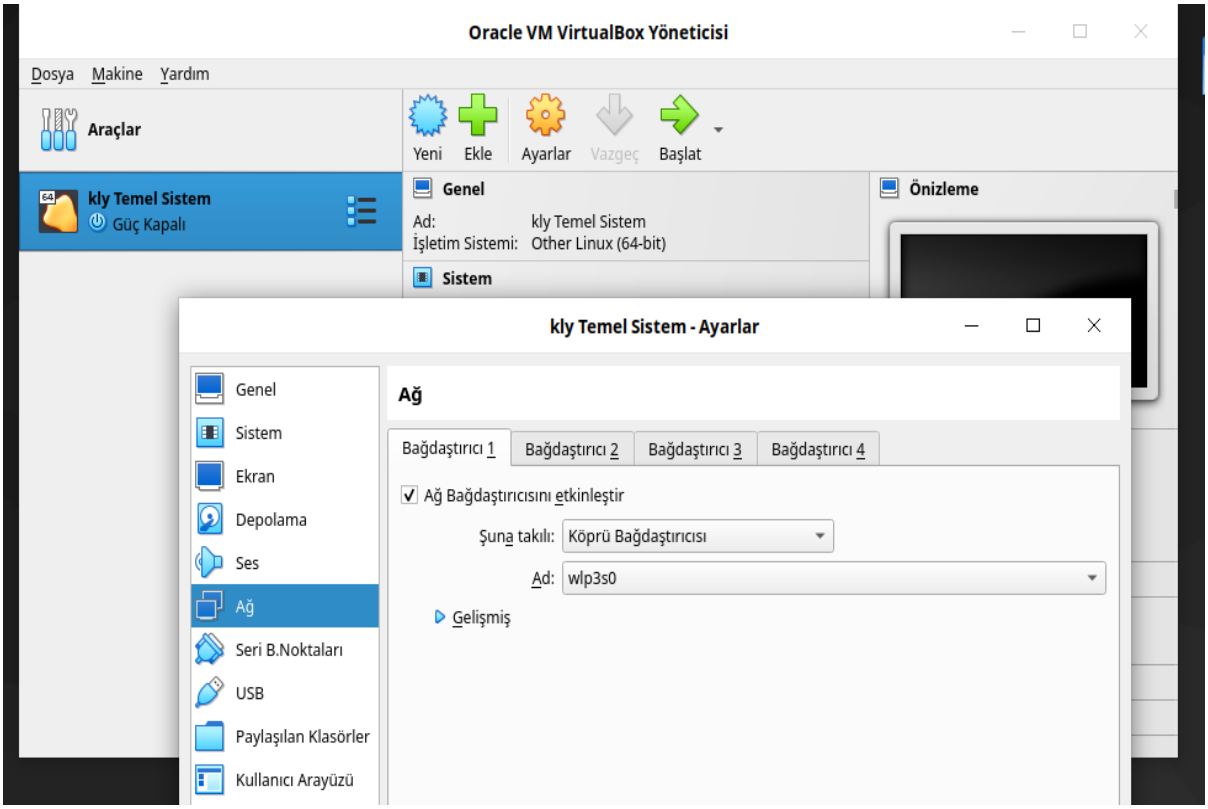
Tüm işlemler bitince aşağıdaki gibi pencere gelir. Bitir'i seçerek işlem tamamlanır.



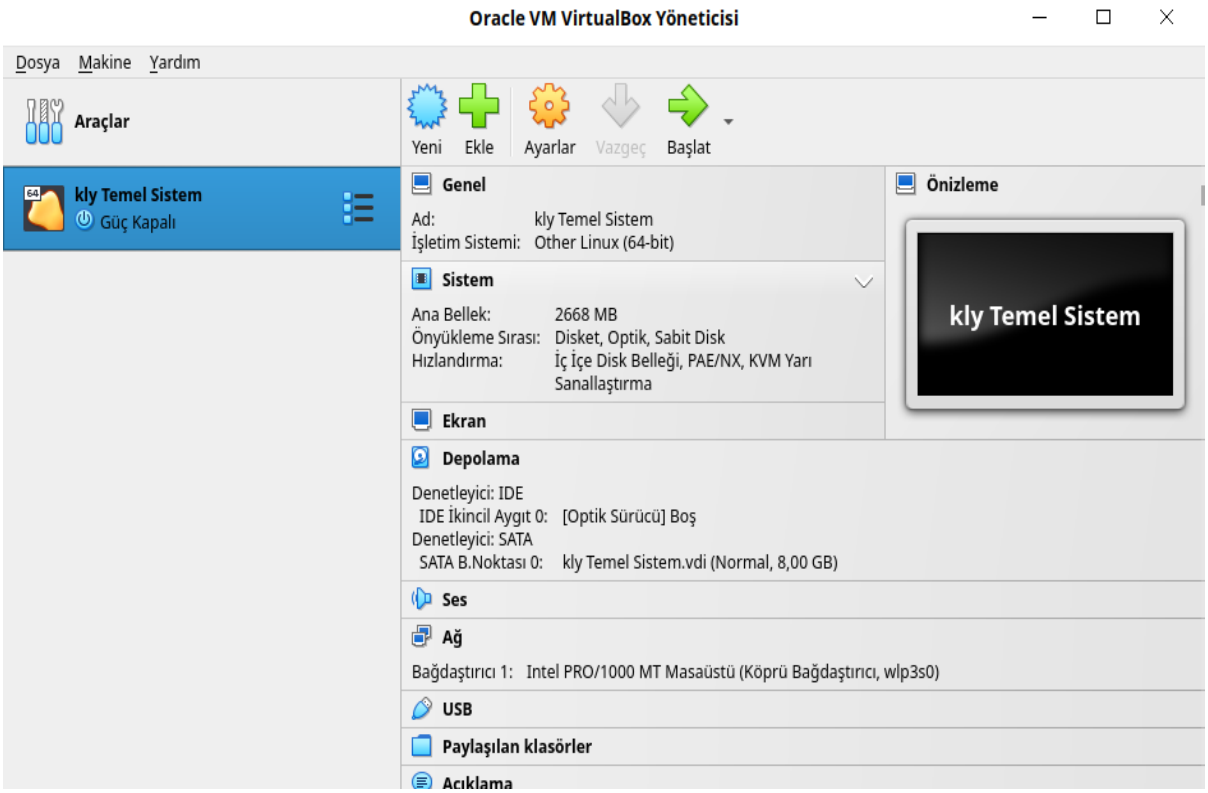
Bitir işlemi seçildikten sonra aşağıdaki gibi görünecektir.



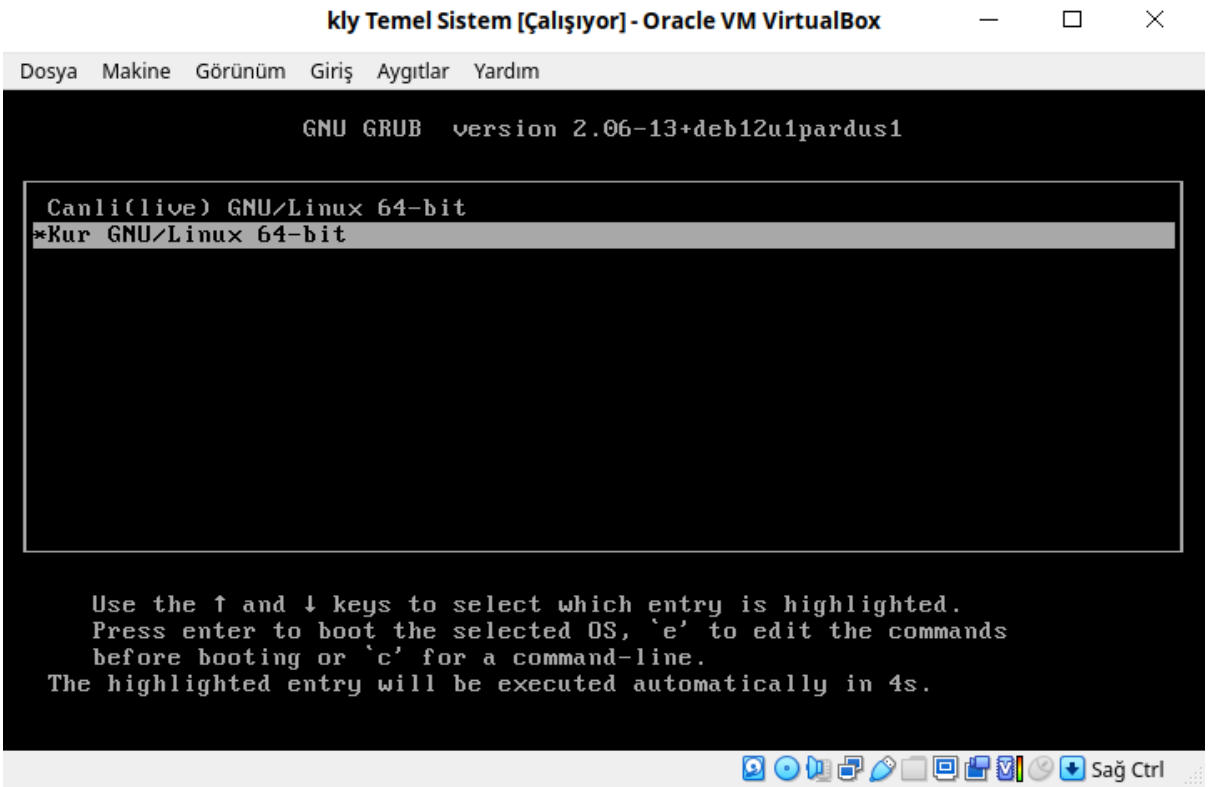
Ayarlar bölümünde aşağıdaki gibi **Ağ** ayarında değişiklik yapılır.



Başlat seçilerek sistem çalıştırılır.



İso açıldığıda aşağıdaki gibi ekran gelecektir. Burada seçili olan iso **Temel Sistem** isosudur.



Live Sistem Oluşturma ve Sistem Kurulumu

Canlı sistem oluşturma veya RAM üzerinden çalışan sistem hazırlamak için SquashFS dosya sisteminde dağıtım sıkıştırılmalıdır. SquashFS, Linux işletim sistemlerinde sıkıştırılmış bir dosya sistemidir. Sistemimizi sıkıştırır ve ardından salt okunur bir dosya sistemine dönüştürür.

SquashFS Oluşturma

```
# mksquashfs input_source output/filesystem.squashfs -comp xz -wildcards
mksquashfs initrd $HOME/distro/filesystem.squashfs -comp xz -wildcards
```

Cdrom Erişimi

Cd veya Dvd aygıtı Linux sistemlerinde /dev/sr0 aygıt dosyası olarak erişilir. Cd içeriği üzerinde okuma yapmak için aşağıdaki komutu kullanabiliriz.

```
cat /dev/sr0
```

Cdrom Bağlama

```
mkdir cdrom
mount /dev/sr0 /cdrom
```

Bu işlem sonucunda cdrom bağlanmış olacaktır. ISO dosyamızın içerisine erişebiliriz.

Squashfs Dosyasını Bulma

Genellikle isoların içine squashfs dosyası oluşturulur. Bu sayede live yükleme yapılabilir. Örneğin, /live/filesystem.squashfs imaj dosyalarında konumudur.

Squashfs Bağlama

Squashfs dosyasını bağlamadan önce loop modülünün yüklü olması gerekmektedir. Eğer yüklemediyseniz;

```
# loop modülü yüklenir.
modprobe loop
mkdir kaynak
mount -t squashfs -o loop cdrom/live/filesystem.squashfs /kaynak
```

Squashfs Sistemine Geçiş

Yukarıdaki adımlarda squashfs dosyamızı **/kaynak** adında dizine bağlamış olduk. Bu aşamadan sonra sistemimizin bir kopyası olan squashfs canlıdan erişilebilir veya sistemi buradan başlatabiliriz.

Squashfs dosya sistemimize bağlanmak için;

```
chroot canli /bin/bash
```

Bu işlemin yerine exec komutuyla bağlanırsak sistemimiz id "1" değeriyle çalıştıracaktır. Eğer sistemin bu dosya sistemiyle açılmasını istiyorsak exec ile çalıştırıp id=1 olmasına dikkat etmeliyiz.

Bu bölümde **/kaynak** bağlanan sistemin disk üzerine kopyalanması ve sisk üzerinden sistemi açacak şekilde hazırlanmasını iki senaryo için anlatılmıştır.

1-Tek Bölüm Kurulumu

Hazırlıklar:

- Disk işlemleri için *evdev* veya *udevd* servislerinin aktif olması gerekir.
- Gerekli çekirdek modüllerini yükleyin:

```
modprobe loop
modprobe squashfs
modprobe ext4
```

Disk Bölümlendirme:

```
cfdisk /dev/sda
```

1. *dos* seçilmeli
2. *type linux system* seçilmeli
3. *write* yazılmalı
4. *quit* yapılmalı
5. Bu işlem sonucunda sadece **/dev/sda1** bölümü olacaktır

Disk biçimlendirme:

```
$ mkfs.ext2 /dev/sda1
```

Dosya Sistemi Oluşturma:

```
mkfs.ext4 /dev/sda1
```

Kurulum Medyasını Bağlama ve Dosya Kopyalama:

```
mkdir -p /cdrom /kaynak
mount -t iso9660 -o loop /dev/sr0 /cdrom
mount -t squashfs -o loop /cdrom/live/filesystem.squashfs /kaynak
mkdir -p /hedef
mount /dev/sda1 /hedef
cp -prfv /kaynak/* /hedef
sync
```

Chroot Ortamına Geçiş ve GRUB Kurulumu:

```
for dir in /dev /sys /proc /run /tmp; do
    mkdir -p /hedef/$dir
    mount --bind /$dir /hedef/$dir
done

chroot /hedef
grub-install --boot-directory=/boot /dev/sda
grub-mkconfig -o /boot/grub/grub.cfg
```

2-UEFI Sistem Kurulumu

Hazırlıklar:

- Disk işlemleri için *evdev* veya *udevd* servislerinin aktif olması gerekir.
- Gerekli çekirdek modüllerini yükleyin:

```
modprobe loop
modprobe squashfs
modprobe ext4
```

Disk Bölümlendirme:

```
cfdisk /dev/sda
```

1. *gpt* seçin
2. 512MB *uefi* alanı oluşturun (sda1)
3. Geri kalan alanı *Linux system* olarak ayarlayın (sda2)
4. *Write* ve ardından *Quit*
5. Bu işlem sonucunda sadece **/dev/sda1 ve dev/sda2** bölümü oluşmalı.

Bölümleri biçimlendirme:

```
mkfs.vfat /dev/sda1
mkfs.ext4 /dev/sda2
```

Kurulum Medyasını Bağlama ve Dosya Kopyalama:

```
#-----------------------------------------------------------------------------------------------------------------
mkdir -p /cdrom /kaynak
mount -t iso9660 -o loop /dev/sr0 /cdrom
mount -t squashfs -o loop /cdrom/live/filesystem.squashfs /kaynak
mkdir -p /hedef
mkdir -p /hedef/boot/efi
mount /dev/sda2 /hedef
mount /dev/sda1 /hedef/boot/efi
cp -prfv /kaynak/* /hedef
sync
```

Chroot Ortamına Geçiş ve GRUB Kurulumu:

```
for dir in /dev /sys /proc /run /tmp; do
    mkdir -p /hedef/$dir
    mount --bind /$dir /hedef/$dir
done

chroot /hedef

mount -t efivarfs efivarfs /sys/firmware/efi/efivars
grub-install --removable --boot-directory=/boot --efi-directory=/boot --target=x86_64-efi /dev/sda
grub-mkconfig -o /boot/grub/grub.cfg
```

Kaynak:

```
- https://wiki.archlinux.org/title/GRUB_(T%C3%BCrk%C3%A7e)#UEFI_sistemler - 08/07/2025
- https://tldp.org/HOWTO/html_single/SquashFS-HOWTO/ - 09/07/2025
- https://askubuntu.com/questions/437880/extract-a-squashfs-to-an-existing-directory - 11/07/2025
- https://grok.com/share/c2hhcmQtMw%3D%3D_2ad2ac6b-d067-40b0-9b55-46db0c2c98dc - 10/07/2025
```

kmod Nedir?

Linux çekirdeği ile donanım arasındaki haberleşmeyi sağlayan kod parçalarıdır. Bu kod parçalarını kernele eklediğimizde kerneli tekrardan derlememiz gerekmektedir. Her eklenen koddan sonra kernel derleme, kod çıkarttığımzda kernel derlemek ciddi bir iş yükü ve karmaşa yaratacaktır.

Bu sorunların çözümü için modul vardır. moduller kernele istediğimiz kod parpalarını ekleme ya da çıkartma yapabiliyoruz. Bu işlemleri yaparken kenel derleme işlemi yapmamıza gerek yok.

Kernele modul yükleme kaldırma için kmod aracı kullanılmaktadır. kmaod aracı;

```
ln -s kmod /bin/depmod
ln -s kmod /bin/insmod
ln -s kmod /initrd/bin/lsmod
ln -s kmod /bin/modinfo
ln -s kmod /bin/modprobe
ln -s kmod /bin/rmmod
```

şeklinde sembolik bağlarla yeni araçlar oluşturulmuştur.

lsmod : yüklü modulleri listeler

insmod: tek bir modul yükler

rmmod: tek bir modul siler

modinfo: modul hakkında bilgi alınır

modprobe: insmod komutunun aynısı fakat daha işlevseldir. module ait bağımlı olduğu modülleride yüklemektedir. modprobe modülü /lib/modules/ dizini altında aramaktadır.

depmod: /lib/modules dizinindeki modüllerin listesini günceller. Fakat başka bir dizinde ise basedir=konum şeklinde belirtmek gerekir. konum dizininde /lib/modules/** şeklinde kalsörler olmalıdır.

strip

strip komutu, derlenmiş program veya paylaşılan kütüphanelerin dosya boyutunu küçültmek için kullanılır. Derleme sırasında eklenen gereksiz sembol ve hata ayıklama bilgilerini kaldırarak dosyayı küçültür.

strip komutunu kullanmak için terminalde şu komutu yazabilirsiniz:

```
strip dosya_adı
```

Burada "dosya_adı", boyutu azaltılacak dosyanın adıdır. Örneğin:

```
strip program
```

Bu komut, özellikle Linux ve Unix sistemlerinde yaygın kullanılır. Dosya boyutunu küçülterek disk alanı tasarrufu sağlar ve programların dağıtımı için uygundur.

Ancak, hata ayıklama veya sembol bilgilerine ihtiyaç duyduğunuz durumlarda strip komutunu kullanmamanız gerekir.

Özetle, strip komutu derlenmiş dosyaların gereksiz bilgilerini temizleyerek boyutlarını azaltan basit ve etkili bir Linux aracıdır.

Kullanıcı Ekleme

linux sistemlerine kullanıcı eklemek için iki farklı komut bulunur. Bu komutlar adduser ve useradd komutlarıdır.

adduser:

Kullanıcı dostu bir arayüz sunar. Birçok aşamayı bizim adımıza yapar. Temelde useradd komutunu kullanarak yazılan bir script olarak düşünebiliriz.

```
# adduser komutuyla kullanıcı oluşturma
sudo adduser yeni_kullanici
```

useradd:

Kullanıcıdan tüm parametreleri girmesini ister. Bu sebepten çok tercih edilmemektedir. Fakat özel bir şekilde kullanıcı oluşturulmak istenildiğinde tercih edilir. Bu dokümanda kurulum aşamalarında useradd komutu kullanıldı.

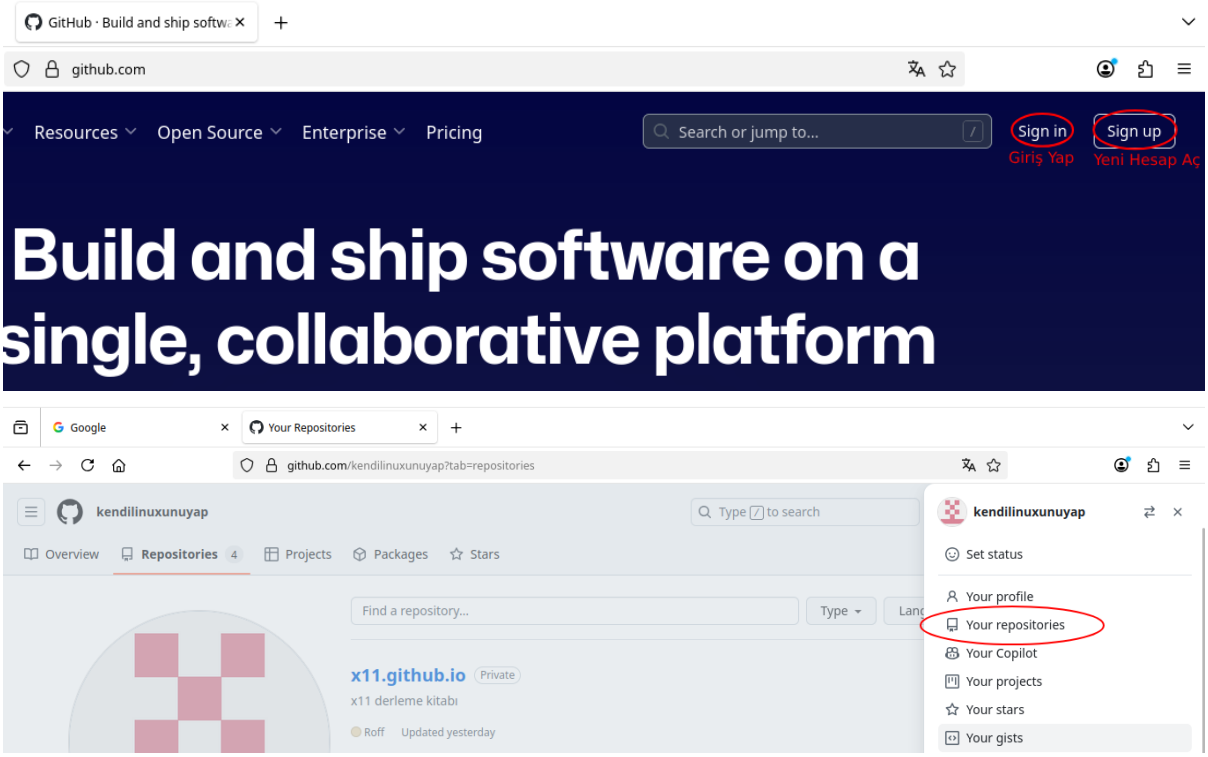
```
# useradd komutuyla kullanıcı oluşturma
sudo useradd -m -s /bin/bash yeni_kullanici -d /home/yeni_kullanici
# -s/bin/bash : kullanıcının kullanacağı shell belirtiliyor.
# -d /home/yeni_kullanici : home dizinine oluşturulacak kullanıcı dizini belirtiyoruz
```

github

GitHub üzerinde yeni bir depo açmak oldukça basit bir işlemdir. Aşağıdaki adımları takip ederek hızlıca kendi deponuzu oluşturabilirsiniz:

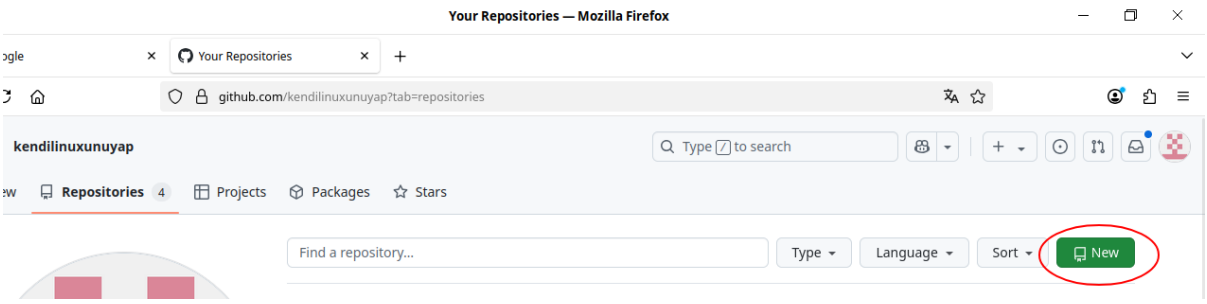
GitHub Hesabınıza Giriş Yapın

GitHub ana sayfasına gidin ve hesabınıza giriş yapın. Eğer bir hesabınız yoksa, öncelikle bir hesap oluşturmalısınız.



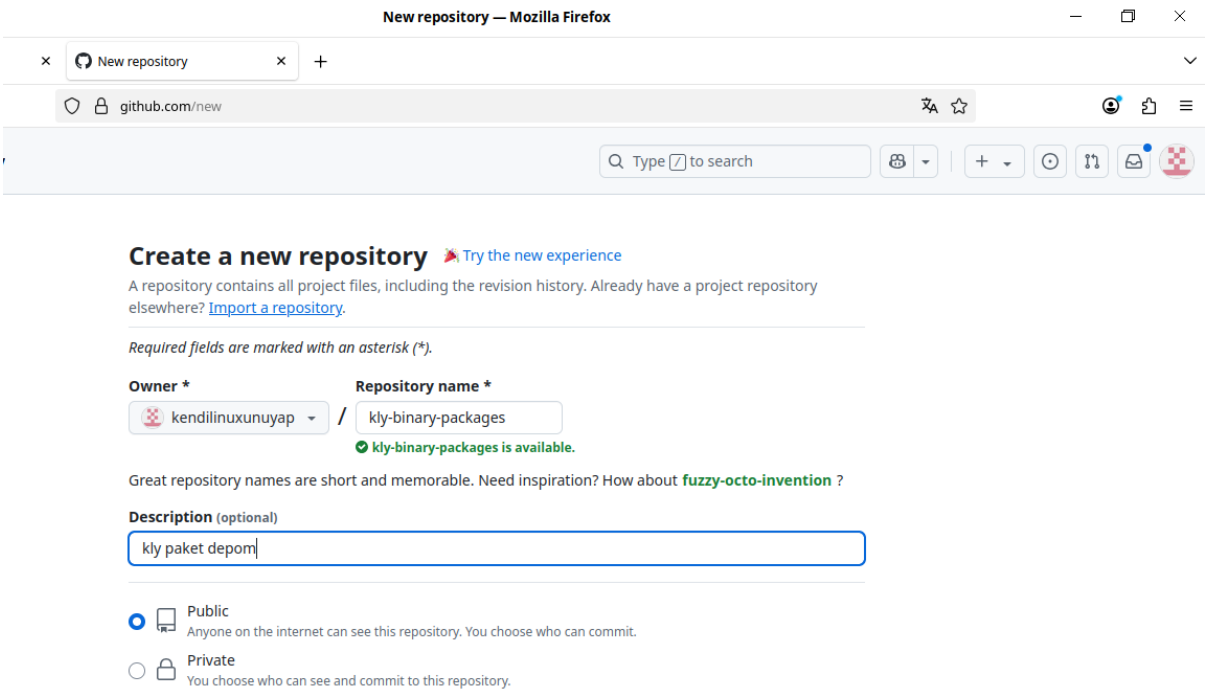
Yeni Depo Oluşturma

Sağ üst köşede bulunan "+" simgesine tıklayın ve "New repository" seçeneğini seçin.



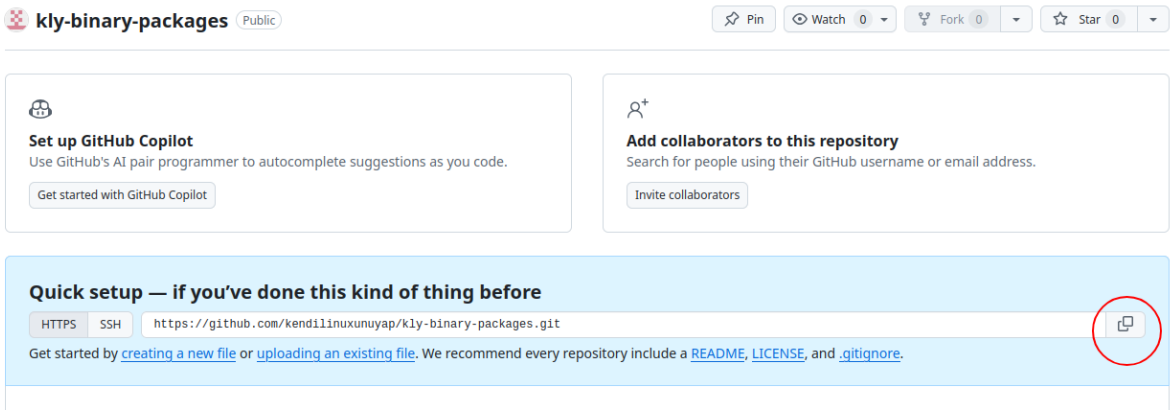
Depo Bilgilerini Girin

Açılan sayfada, depo adını (repository name) ve isteğe bağlı olarak bir açıklama (description) girin. Depo özel (private) veya herkese açık (public) olarak ayarlanabilir.



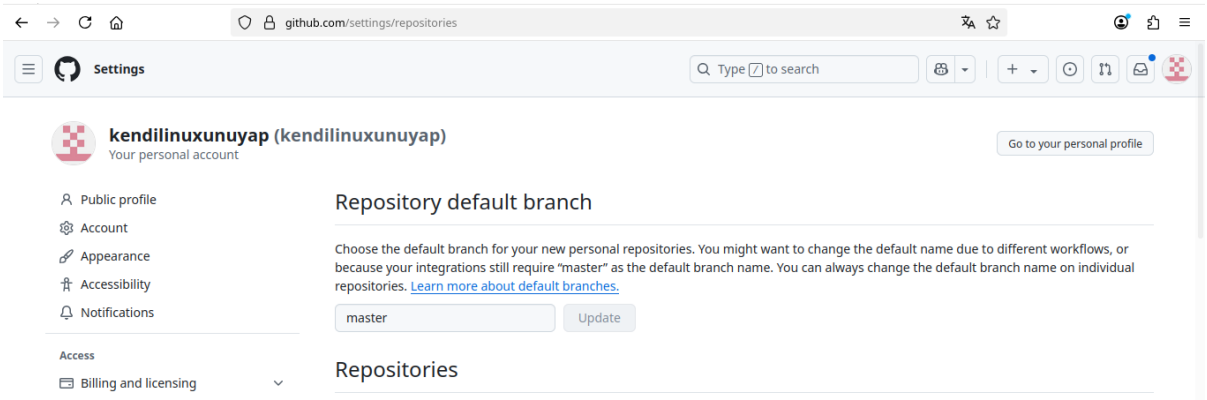
İlk Dosyayı Oluşturma

"Initialize this repository with a README" seçeneğini işaretleyerek, depo oluşturulduğunda otomatik olarak bir README dosyası oluşturabilirsiniz.

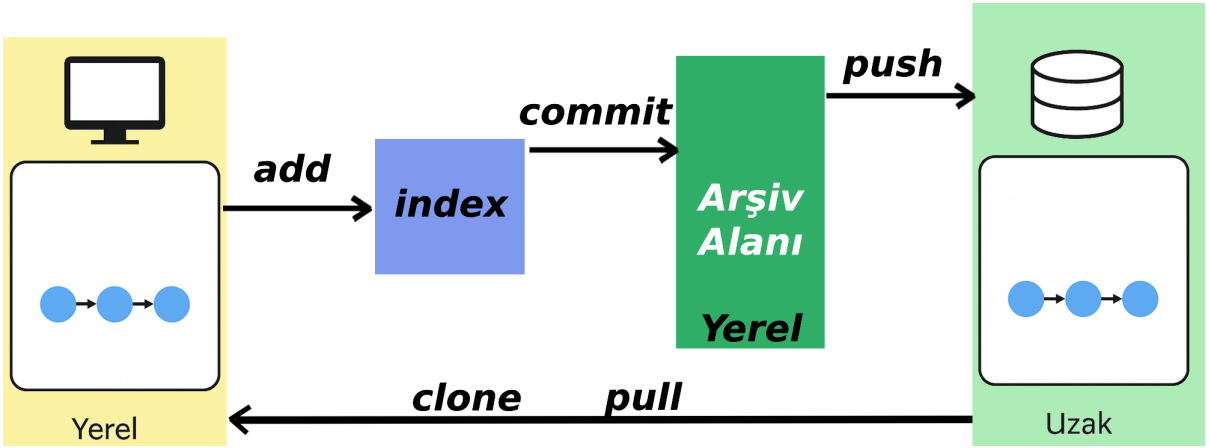


github Varsayılan Dal Ayarı:

githubda varsayılan olarak eskilede master, yeni sürümlerde main kullanılmaktadır. Projelermizde ve burad kullanılan yapılarda **master** kullanıldığı için aşağıda görülduğü gibi varsalılan dalı **master** yapıyoruz.



github komut Kullanımı



github'ın çalışma mantığı yukarıda verilen resimde görüldüğü gibidir. Komutları kullanırken resimdeki gibi işlemleri yapmalıyız.

github'a göndermek için; **add --> commit --> push** kullanmalıyız.

github'dan indirmek için; **clone veya pull** kullanmalıyız.

Sık Kullanılan github Komutları

Depoyu Yerele indirme(Clone)

```
git clone https://github.com/kullaniciadi/depoadi.git
```

- **Yerel Depoda Dosya Ekleme:**

```
git add .
```

- **Yerel Depoda Değişiklik Etiketi Yapma:**

```
git commit -m "ilk adım"
```

- **Yerel Depodaki Bilgileri Guthuba Gönderme:**

```
git push origin master
```

- **Yerel Depodaki Bilgileri Guthuba Gönderme Reddedilirse:**

```
git push origin master --force
```

- **Yerelde github'daki Depoyu clone Yapmadan Oluşturma:**

```
cd proje
git init
git config --global user.name "name"
git config --global user.email "name@gmail.com"
git add README.md
git add .
git commit -m "first commit"
git remote -v      # push ve pull yapılacak adresleri görmek için kullanılır
git remote add origin https://github.com/userName/repoName.git
git push -u origin master
```

Dal(Branch):

Dal projenin birden fazla kişi ile yapılmasında veya yeni özellikler eklenmek istediğinde projenin bir kopyası ile çalışma gerektirir. Aşağıda dal işlemleri içinkomutlar verilmiştir.

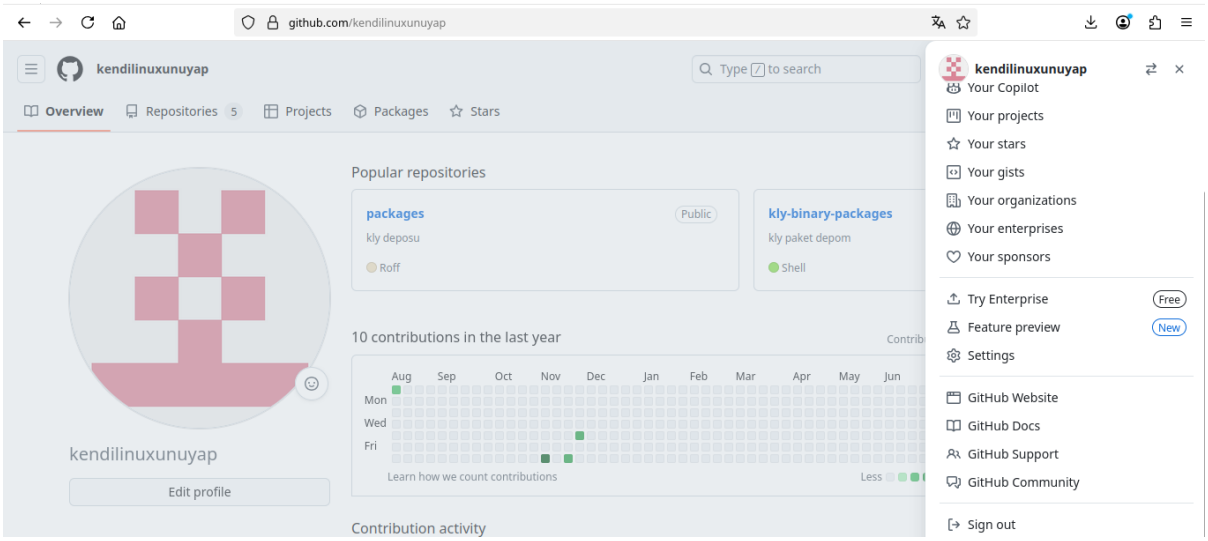
Yeni Dal Oluşturmak, Seçmek ve Yeni Dalı github'a Göndermek:

```
# Dalları Görmek kullanılır Seçili olan dalın rengi farklı olur ve önünde * olur
git branch
# Dal oluşturmak
git checkout yeni
# Yeni dalı uzak adrese göndermek
git push --force origin master komutu  yerine
# Uzak adresimizde master dalı dışında yeni dalımızda oluşacaktır...
git push --force origin yeni komutunu veriyoruz.
```

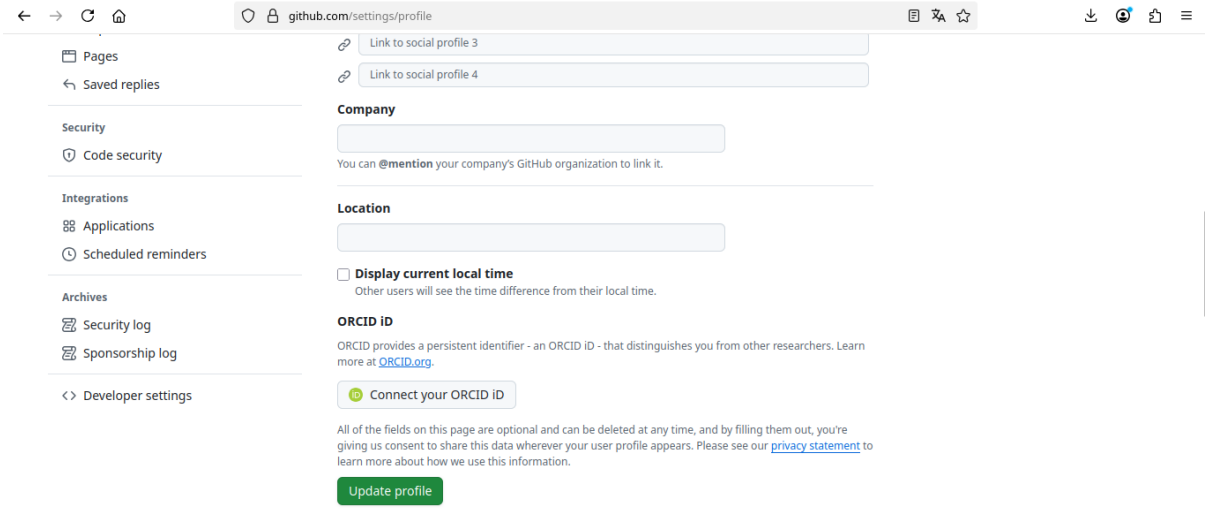
github token Oluşturma Kullanma

github kullanırken dosya gondermek(**commit**) için kullanıcı adı ve parola ister. Burada hesap parolası yerine **token** kullanılır. Yeni bir **token** için aşağıdaki işlem adımlarını yapmalıyız.

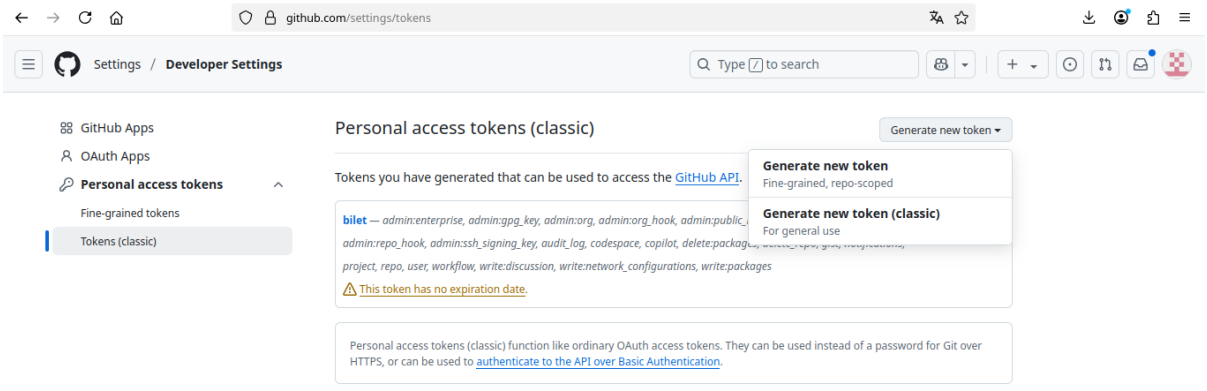
Settings Seçilir;



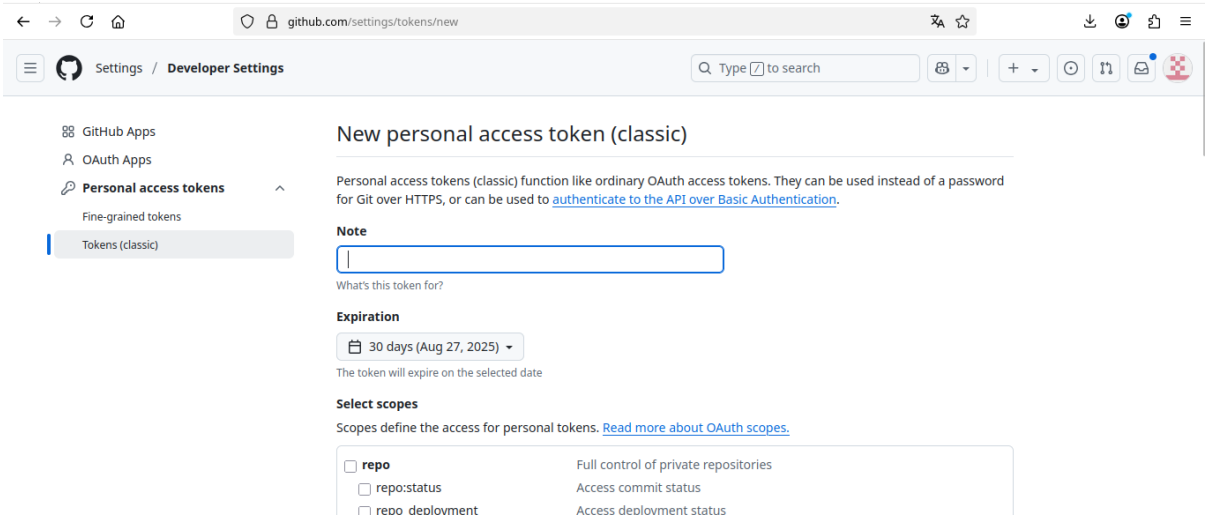
Developper Settings Seçilir;



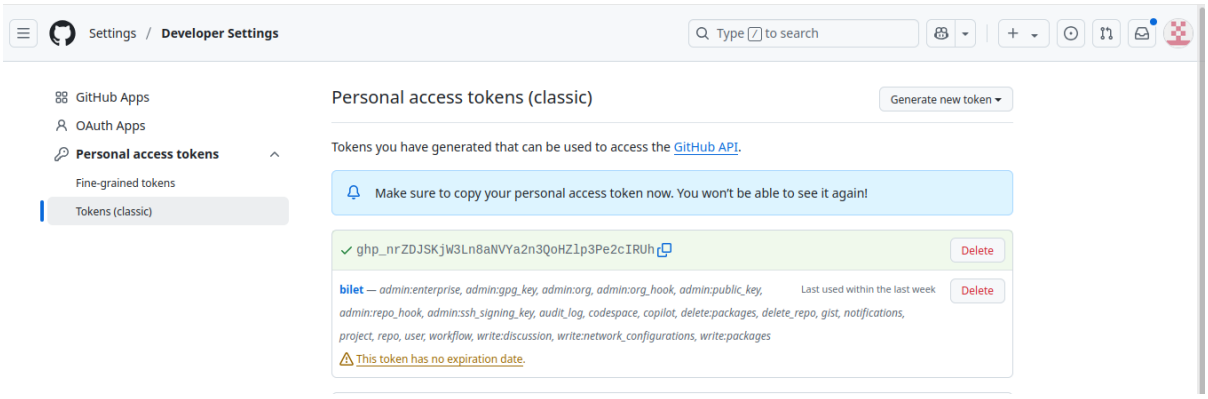
Generate New Token Seçilir;



Erişim yapabileceği alanlar Seçilir;



token Kullanmak üzere kullanmak üzere saklanır. Bu ekrandan sonra sadece silebiliriz. Göremeyiz kopyalayamayız.



X11 Kullanıcısının Tespiti

Linux ortamında masaütündeyken sudo ile script çalıştırıldığında masaütünü açtığımız kullanıcının tespiti aşağıdaki kodla yapılabilir.

```
#!/bin/bash

# Display adı
display=":$(ls /tmp/.X11-unix/* | sed 's#/tmp/.X11-unix/X##' | head -1)"

# Display kullanıcısı
user=$(who | grep "(${display})" | awk '{print $1}')

echo $user $display
```

Kaynaklar


```
- https://tr.wikipedia.org/wiki/Linux - 02/07/2025
- https://www.subrat.info/build-kernel-and-userspace/ - 08/07/2025
- https://medium.com/@chienhaotan/compiling-and-running-a-minimal-kernel-with-busybox-bfc45a991017 - 08/07/2025
- https://stackoverflow.com/questions/64838052/how-to-delete-n-characters-appended-to-ldd-list - 20/06/2025
- https://gist.github.com/bluedragon1221/a58b0e1ed4492b44aa530f4db0ffef85 - 09/07/2025
- https://app.diagrams.net/
- https://www.ubuntubuzz.com/2021/04/how-to-boot-uefi-on-qemu.html - 20/06/2024
- https://wiki.gentoo.org/wiki/OpenRC - 30/06/2025
- https://busybox.net/  - 01/07/2025
- https://busybox.net/about.html - 01/07/2025
- https://sulincix.gitlab.io/distro-kitabi/ - 05/07/2025
- https://gitlab.com/turkman/devel/doc/wiki/ - 05/07/2025
- https://turkman.gitlab.io/devel/doc/wiki/ - 05/07/2025
- https://grok.com/share/c2hhcmQtMw%3D%3D_5c049e44-be57-4652-872f-55def669d917 - 09/07/2025
- https://www.linuxfromscratch.org/lfs/
- https://wiki.archlinux.org/title/GRUB_(T%C3%BCrk%C3%A7e)#UEFI_sistemler - 08/07/2025
- https://tldp.org/HOWTO/html_single/SquashFS-HOWTO/ - 09/07/2025
- https://askubuntu.com/questions/437880/extract-a-squashfs-to-an-existing-directory - 11/07/2025
- https://grok.com/share/c2hhcmQtMw%3D%3D_2ad2ac6b-d067-40b0-9b55-46db0c2c98dc - 10/07/2025
- https://chatgpt.com/
```


Not: Metin düzenlemelerinde chatgpt kullanılmıştır.

Geliştiricilere Mesajımız

Gnu/Linux, açık kaynaklı bir işletim sistemidir. Kullanıcı ve geliştiriclere bir çok avantajlar sunmaktadır. Bunlar;

- Kodlarına erişilebilir(Açık Kaynak)
- Dağıtılabir
- Değiştirilebilir
- Virüsten etilenme azdır
- Özelleştirilebilir
- Bağımlılığı azaltır
- Güvenlik en önemli şarttır
- Düşük donanımlarda iyi performan verir

Bu özellikleri açısından Gnu/Linux tercih etmek çok avantajlıdır. Geliştirme aşamalarına destek vermek, öğrenmek bize, toplumumuza ve ülkemize sayısız faydalar saylayacaktır.

Lisans Bilgileri

1. **Kaynak Kodlar (Source Code)**:

   Bu dokümandaki kaynak kodların tamamı Free Software Foundation tarafından yayınlanan GNU Genel Kamu Lisansı'nın (GPL) 3. versiyonu ile lisanslıdır.

   Lisansın bir kopyasını şu adresten edinebilirsiniz: https://www.gnu.org/licenses/gpl-3.0.html

İletişim

- https://github.com/kendilinuxunuyap
- https://kendilinuxunuyap.github.io
- kendilinuxunuyap@gmail.com

ISBN: 978-625-00-6004-9

Yayın Yılı: Nisan 2026