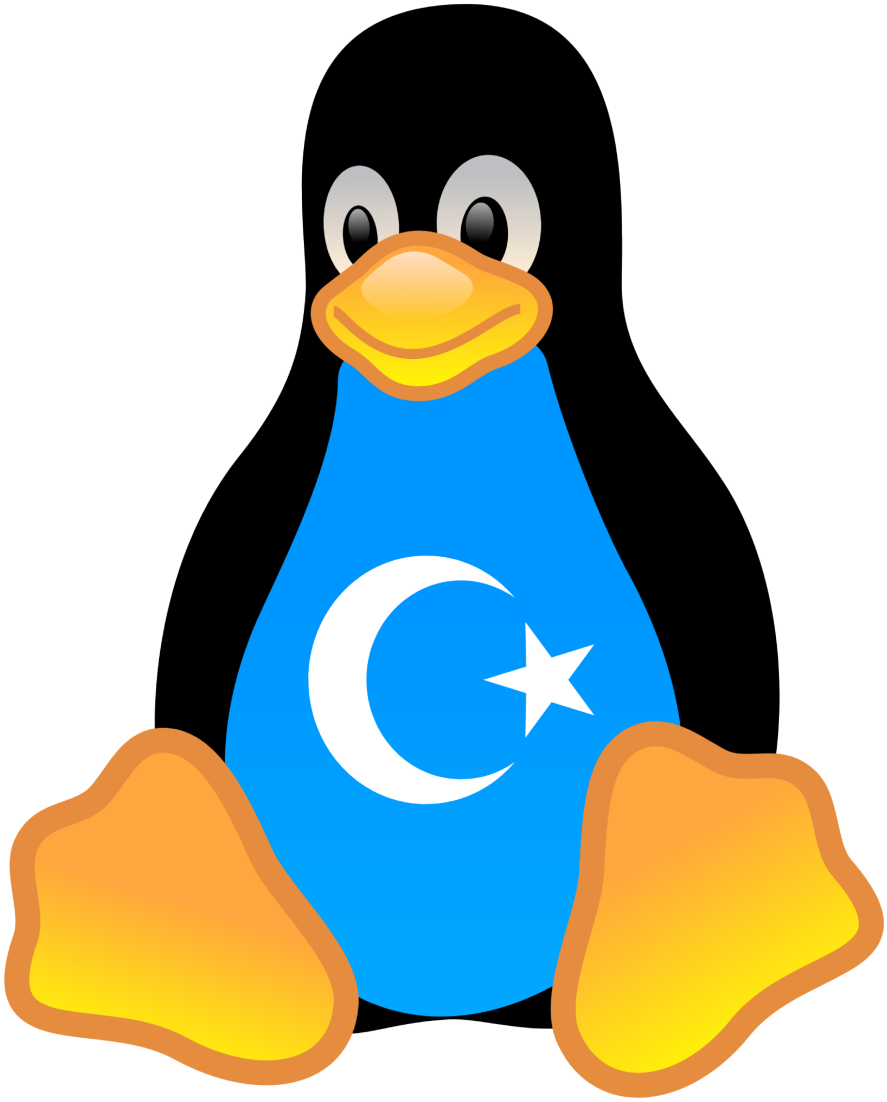
KENDİ LİNUX'UNU YAP



Yazar

- Bayram KARAHAN
- Celalettin AKARSU
- Ali Rıza KESKİN

Paket Derleme

- Bayram KARAHAN
- Celalettin AKARSU
- Ali Rıza KESKİN

Ön Söz

Bu kitap, açık kaynak ve özgür yazılıma gönül vermiş kişilerin, Türkçe kaynak bulamama sıkıntısına bir çözüm olarak hayata geçirilmiş bir projedir. Açık kaynak dünyasında kendi dağıtımlarını yapmak isteyen kişilere bir rehber olacak şekilde hazırlanmıştır. Bu dokümanda anlatılan sıralamaya göre işlemleri uyguladığınızda kendi sisteminiz derleyip iso halinde son kullanıcını kullanabileceği gibi bir sistem ortaya çıkartabilirsiniz.

İçindekiler

Temel Kavramlar

Temel Kavramlar

Linux Nedir?

Linux; Geliştirilmesi Linus Torvalds tarafından başlatılmış açık kaynaklı bir işletim sistemidir. GPL2(Genel Kamu Lisansı)'ye bağlı kalarak geliştirilmesi devam etmektedir. Linux bilgisayarlar, tablet, telefon ve birçok cihazda kullanılmaktadır. Linux çekirdeği kullanılarak oluşturulan sistemlere genellikle Linux denilmektedir. Doğrusu GNU/Linux ifadesidir.

GPL Nedir?

Yazılımları özgür şekilde kullanmak için lisans şartlardıdır.

GNU Nedir?

GNU yazılımlara özgürce kullanmayı hedefyen işbirliğini temel alan özgür yazılım tasarısıdır. Kısaca yazılımı al, kullan, değiştir dağıt gibi felsefeyi benimsemiştir. Linux kerneli GNU GPLv2 lisansıyla dağıtılmakta ve kullanılmakdtadır. GNU açılımı (GNU Unix değildir). Günümüzde kullanığımız linux işletim sistemi diye bilinen sistemler aslında GNU/Linux işletim sistemidir.

Açık Kaynak Nedir?

Açık kaynak, geliştirilen uygulamanın kodlarının herkes tarafından ulaşılabildiği ve GPL vb. lisanlara bağlı kalarak kullanabildiği kodlar bütünüdür. Kodlar bir çok yazılımcı tarafından erişildiiği ve kullanıldığı için defalarca kontrol edilmekte, hatalar tespit edilerek geliştiricisiine geri dönüt sağlamaktadır. Bu durum çok hızlı şekilde hatasız ve güvenilir kodlar oluşmaktadır. Genellikle github, gitlab benzeri ortamlarda paşlaşılmaktadır.

Dağıtım Nedir ve Türleri

GNU/Linux, kerneli, GNU uygulamaları ve yardımcı kütühanelerin bir araya getirilmesiyle olşuturulan uygulamalar bütününe dağıtım denilmektedir. Genellikle bu oluşan yapıya doğru bir ifade olmasada Linux denilmekte. Dağıtımların aşağıdaki gibi türleri kullanılmaktadır.

1. Önceden derlenmiş Linux Kerneli, GNU uygulamalarını bir araya getirip dağıtım oluşturma.
2. Hazır bir dağıtımı değiştirip dağıtma
3. Paket sistemi olan dağıtım oluşturup dağıtma

Yazılım Bağımlılığı

Herkes tarafından erişilebilen ve lisanslarına bağlı kalarak kullanılabilen, sahiplenilebilen yazılımlar bağımsız uygulamalar geliştirmemizi sağlar. Bu özellikleri sağlayan yazılımlar açık kanak yazılımlarıdır. Açık kaynak yazılımlar sayesinde geliştiriciler, şirketler, ülkeler bir kurum veya kuruluşa bağımlı oldan yazılım ihtiyaçlarını karşılayabilmektedir. Bundan dolayı yazılım bağımsızlığı maliyet, güvenlik, ağır lisans şatları ve yeni yazılımları geliştirme engellerini büyük ölçüde kaldırmaktadır.

Ön Hazırlık

Dağıtım Ortamın Hazırlanması

Dağıtım hazırlarken sistemin derlenmesi ve gerekli ayarlamaların yapılabilmesi için bir linux dağıtımı gerekmektedir. Tecrübeli olduğunuz bir dağıtımı seçmeniz uygun olur. Fakat seçilecek dağıtım Gentoo olması daha hızlı ve sorunsuz sürece devam etmenizi sağlayacaktır. Bu dağıtımı hazırlaken Debian dağıtımı kullanıldı. Bazı paketler için, özellikle bağımlılık sorunları yaşanan paketler için ise Gentoo kullanıldı.

Bir dağıtım hazırlamak için çeşitli paketler lazımdır. Bu paketler;

- **debootstrap :** Dağıtım hazırlarken kullanılacak chroot uygulaması bu paket ile gelmektedir. chroot ayrı bir konu başlığıyla anlatılacaktır.
- **make :** Paket derlemek için uygulama
- **squashfs-tools:** Hazırladığımız sistemi sıkıştırılmış dosya halinde sistem görüntüsü oluşturmamızı sağlayan paket.
- **gcc :** c kodlarımızı derleyeceğimiz derleme aracı.
- **wget :** tarball vb. dosyaları indirmek için kullanılacak uygulama.
- **unzip :** Sıkıştırmış zip dosyalarını açmak için uygulama
- **xz-utils :** Yüksek sıkıştırma yapan sıkıştırma uygulaması
- **tar :** tar uzantılı dosya sıkıştırma ve açma içiçn kullanılan uygulama.
- **zstd :** Yüksek sıkıştırma yapan sıkıştırma uygulaması
- **grub-mkrescue :** Hazırladığımız iso dizinini iso yapmak için kullanılan uygulama
- **qemu-system-x86:** iso dosyalarını test etmek ve kullanmak için sanal makina emülatörü uygulaması.

```
sudo apt update
sudo apt install debootstrap xorriso mtools make squashfs-tools gcc wget unzip xz-utils tar zstd -y
```

Paket kurulumu yapıldıktan sonra dağıtımı hazırlayacağımız yeri(hedefi) belirlemeliyiz. Bu dokümanda sistem için kurulum dizini $HOME/distro/rootfs olarak kullanacağız.

$HOME: Bu ifade linux ortamında açık olan kullanıncının ev dizinini ifade eder. Örneğin sisteme giriş yapan kullanıcı **ogrenci** adinda bir kullanıcı olsun. **$HOME** ifadesi **/home/ogrenci** konumunu ifade eder.

Derleme Türleri

Derleme Türleri

Kaynak kodları elimizde olan bir uygulama dosyası **dynamic** ve **static** olarak iki farklı şekilde derlenebilir. Derlenecek uygulama kodu **main.c** içeriği aşağıdadır.

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

Dynamic Derleme

Derlenen uygulama sistemde bulunan kütüphaneleri kullanacak şeklide derlenmesidir. Aşağıdaki gibi derlenir.

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

Derlediğimiz uygulamanın kullandığı kütüphaneleri **ldd** komutu kullanarak öğrenilir.

```
unset LD_PRELOAD
ldd ./main
        linux-vdso.so.1 (0x00007fff915fe000)
        libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007f622270f000)
        /lib64/ld-linux-x86-64.so.2 (0x00007f6222917000)
```

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]
```

interpreter Kavramı

libc.so.6 ve ld-linux-x86-64.so.2 glibc tarafından sağlanır derlenmiş dosyaların çalışması için gereklidir. Bu dosyalardan libc.so.6 temel C kütüphanesidir. ld-linux-x86-64.so.2 ise interpreter olup dosyanın ne şekilde çalıştırılacağını belirler. Bir dosyanın hangi interpreter ile çalıştığını bulmak için file komutuyla görebiliriz. Aşağıdaki file çıktısında **interpreter /lib64/ld-linux-x86-64.so.2** olduğunu görüyoruz. linux-vdso.so.1 ise kernel tarafından sağlanır ve herhangi bir dosya şeklinde bulunmaz.

```
file main
main: ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV), dynamically linked,
interpreter /lib64/ld-linux-x86-64.so.2, BuildID[sha1]=97ab2d8962f9a56a6a29ae7169cf9692d06956e7,
for GNU/Linux 3.2.0, not stripped
```

LD_LIBRARY_PATH

Derlenmiş bir dosyanın bağımlılığı genellikle sistemde kurulu bulunmalıdır. Bazen farklı konumda özel bir kütüphane gerekebilir. Bu durumlarda LD_LIBRARY_PATH çevresel değişkenini kullanarak konum belirtilir.

```
ldd main
linux-vdso.so.1 (0x00007fff915fe000)
libmain.so => not found
libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007f622270f000)
/lib64/ld-linux-x86-64.so.2 (0x00007f6222917000)
```

libmain.so dosyası sistemdeki yeri bilinmediğinden hata aldık. Şimdi **LD_LIBRARY_PATH** ile konumunu belirtelim, tekrar deneyelim.

```
export LD_LIBRARY_PATH=/home/abc/main/libs/
ldd main
linux-vdso.so.1 (0x00007fff915fe000)
libmain.so => /home/abc/main/libs/libmain.so (0x00007f92ed6cb000)
libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007f622270f000)
/lib64/ld-linux-x86-64.so.2 (0x00007f6222917000)
```

ldd çıktımızda tüm kütüphanelerin bulunduğu görülmektedir.

ldconfig

Sistemdeki kütüpkhanelerin konumlarını **/etc/ld.so.conf** dosyasına bakarak belirler. Aşağıda **/etc/ld.so.conf** içeriğinde sistemde kurulu olan kütüphanelerin konumları yazılmıştır.

```
/usr/local/lib64
/usr/local/lib
include /etc/ld.so.conf.d/*.conf
/usr/lib64
/usr/lib
/lib64
/lib
```

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

Static Derleme

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
```

ldd komutunu kullanarak bağımlı olduğu kütüphaneler varmı diye kontrol edelim. Eğer **static** derlenmişe **not a dynamic executable** mesajı alınır.

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

Static dosyalarda dosyanın çalışması için kütüphanelerin hepsi kendi içerisine gömümülü(dahil) bir şekilde gelir. Avantajı hiçbir kütüphaneye ihtiyaç duymaz. Devevantajı ise boyutları yüksek olur. İstisnalar olsada genel olarak static tercih edilmemektedir.

Derleme Araçları

Derleme Araçları

Linux sistemlerinde kodlarımızı derlemek kullanılan uygulamalara derleme araçları denir. Birden fazla araç olsada en temel derleme araçları aşağıda görülmektedir.

- make
- cmake
- meson
- python

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
```

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 build
    cd build
    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 build
cd build
cmake ..
make
make install
```

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
DESTDIR=/
meson setup builddir --prefix=/usr
ninja -C builddir
DESTDIR=$DESTDIR ninja -C builddir install
```

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.

Minimal Sistem

Minimal Dağıtım Oluşturma

Bu minimal dağıtımı oluşturmamızın amacı bu dokümanda anlatılacak dağıtım hazırlama aşamalarını anlaşılması içindir. Dağıtım oluşturmak için kernel ve busybox dosyalarımızın olması yeterlidir. **busybox** dosyasının nasıl elde edileceği busybox bölümünde anlatılmıştır. **kernel** ise mevcut sistemimin kernelini kullanacağız. kernel **/boot/vmlinux** konumundan alabiliriz.

Bu sistemi hazırlarken **chroot** ve **busybox** komutlarını kullanarak sistemi hazırlayacağız ve test edeceğiz. **chroot** ve **busybox** kullanımı için dokümandanki ilgili konu başlığına bakınız.

Sistemimiz için aşağıdaki gibi bir yapı oluşturmalıyız.

1. $HOME/distro/rootfs/busybox
2. $HOME/distro/rootfs/init
3. $HOME/distro/iso/boot/initrd.img
4. $HOME/distro/iso/boot/vmlinuz
5. $HOME/distro/iso/boot/grub/grub.cfg
6. $HOME/distro/distro.iso

Linux sisteminin açılabilmesi için 3., 4. ve 5. sıradaki gösterilen konumdaki dosyaların olması yeterli. Bu üç dosyayı yukarıda belirtildiği gibi dizin konumlarına koyduktan sonra, **grub-mkrescue $HOME/distro/iso/ -o $HOME/distro/distro.iso** komutuyla **distro.iso** dosyası elde ederiz. Artık 6. sırada belirtilen iso dosyamız boot edebilen şekilde hazırlanmış olur.

Sistemi oluşturan **3., 4. ve 5.** sırada belirtlen dosyalarımız görevi şunlardır.

$HOME/distro/iso/boot/initrd.img: Dosyasını sistemin açılış sürecinden ön işlemleri yapmak ve gerçek sisteme geçiş sürecini yöneten bir dosyadır. Yazın devamında nasıl hazırlanacağı anlatılacaktır.

$HOME/distro/iso/boot/vmlinuz: Dosyamız kernelimiz oluyor. Ben kullandığım debian sisteminin mevcut kernelini kullandım. İstenirse kernel derlenebilir.

$HOME/distro/iso/boot/grub/grub.cfg: Dosyamız ise initrd.img ve vmlinuz dosyalarının grub yazılımının nereden bulacağını gösteren yapılandırma dosyasıdır.

Bir linux sisteminin açılış süreci şu şekilde olmaktadır.

1. **Bilgisayara Güç Verilmesi**
2. **Bios İşlemleri Yapılıyor(POST)**
3. **LILO/GRUB Yazılımı Yükleniyor(grub.cfg dosyası okunuyor ve vmlinuz ve initrd.img devreye giriyor)**
4. **vmlinuz initrd.img sistemini belleğe yüklüyor**
5. **initrd.img içindeki init dosyasındaki işlem sürecine göre sistem işlemlere devam ediyor**

Yazının devamında sistem için gerekli olan 3 temel dosyanın hazırlanması ve iso yapılma süreci anlatılacaktır. Şimdi sırasıyla dosya konumlarına uygun şekilde dizin ve dosyalarımızı oluşturalım.

Dizinlerin Oluşturulması

Sitemimizi istediğimiz bir yerde terminal açarak aşağıdaki komutları sırasıyla çalıştıralım. Komutları kopyala yapıştır yapabilirsiniz.

```
mkdir $HOME/distro
mkdir $HOME/distro/rootfs
mkdir $HOME/distro/iso
mkdir $HOME/distro/iso/boot
mkdir $HOME/distro/iso/boot/grub
```

$HOME/distro/rootfs/busybox

busybox aşağıdaki komutlarla **$HOME/distro/rootfs/busybox** konumuna kopyalanak sistemimiz için hazır hale getirilir. Bu sistemi debian ortamında hazırlamaktayız. İsterseniz static busybox derleyebilirsiniz. busybox konusu altında anlatılmaktadır. isterseniz derlemeden mevcut sistemin altında bulunan static busybox kullanabiliriz. Eğer yoksa **sudo apt install busybox-static -y** komutuyla debian sistemimize static busybox yükleyebiliriz. Ben debian üzerine yüklediğim busybox'ı kullanacağım. Eğer sistemde yoksa **sudo apt install busybox-static -y** komutuyla yükleyiniz.

```
cd $HOME/distro/rootfs
cp /bin/busybox $HOME/distro/rootfs/busybox
ldd ./busybox
özdevimli bir çalıştırılabilir değil
```

"özdevimli bir çalıştırılabilir değil" dinamik değil diyor yani static kısacası bir bağımlılığı yok demektir. Eğer bağımlılığı olsaydı bağımlı olduğu dosyalarıda konumlarına göre kopyalamamız gerekmekteydi.

$HOME/distro/rootfs/init

Sistem açılırken iki dosyayı arar kernel ve initrd.img dosyası. Bu dosyaları grub.cfg dosyası içinde verilen konumlarına göre arar. Bu dosyaları bulduktan sonra initrd.img dosyasını açar ve içinde init dosyasını arar. init dosyasındaki komutları sırasıyla çalıştırır.

$HOME/distro/rootfs/ konumuna **init** text dosyası oluşturulur(nano, gedit vb.). Aşağıdaki içerik oluşturulan **init** dosyası içine eklenir.

```
#!busybox ash
PATH=/bin
/busybox mkdir /bin
/busybox --install -s /bin
/busybox ash
```

busybox içerisinde linux ortamında kullanılan hemen hemen tüm komutların yerine kullanalılabilir. Bulunduğumuz ortamda **busybox** dışında hiçbir komutun olmadığını düşünün. Mesela **cp** komutuna ihtiyacınız var bu durumda **busybox cp** yazarak kullanılabilirsiniz. Tüm temel komutları busyboxtan türetmek için **/busybox --install -s /bin** komutunu kullanabilirsiniz.

Buradaki komutları sırayla anlatırsak;

1. #!busybox ash: ash bir kabuk veya terminal programıdır.
2. PATH=/bin: komutlar çalışırken nerede arayacağı belirtiliyor
3. /busybox mkdir /bin: **busybox mkdir** yardımıyla **/bin** dizini oluşturuluyor.
4. /busybox --install -s /bin: **/bin** dizinine tüm komutları kısa yol oluşturur.
5. /busybox ash: ash terminalini çalıştırır. Bash yerine alternatif.

kernel, **initrd.img** dosyasını bellek üzerine açıp **init** dosyasını çalıştırınca sırasıyla numaralandırılarak anlattığımız işlemler yapılacak ve çalışabilecek ortam hazırlanacaktır. Daha sonra ash terminali çalışarak kullanıcının kullanımına hazır hale gelecektir.

init çalıştırılabilir yapılır.

```
chmod +x init #komutu ile çalıştırılır yapılır.
```

$HOME/distro/iso/boot/initrd.img

initrd.img dosyası için aşağıdaki komutlar çalıştırılır

```
cd $HOME/distro/rootfs
find ./ | cpio -H newc -o >$HOME/distro/iso/boot/initrd.img
```

Oluşturulan **initrd.img** dosyası çalışacak tty açacak(konsol elde etmiş olacağız). Aslında bu işlemi yapan şey **busybox** ikili dosyası.

$HOME/distro/iso/boot/vmlinuz

```
cp /boot/vmlinuz* $HOME/distro/iso/boot/vmlinuz  #sistemde kullandığım kerneli kopyaladım istenirde kernel derlenebilir.
```

$HOME/distro/iso/boot/grub/grub.cfg

$HOME/distro/iso/boot/grub/ konumuna **grub.cfg** dosyası oluşturun. Aşağıdaki komutları **grub.cfg** dosyası içine eklenir.

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

$HOME/distro/distro.iso

iso oluşturulur.

```
grub-mkrescue $HOME/distro/iso/ -o $HOME/distro/distro.iso # komutuyla iso doyamız oluşturulur.
```

Artık sistemi açabilen ve tty açıp bize suna bir yapı oluşturduk.

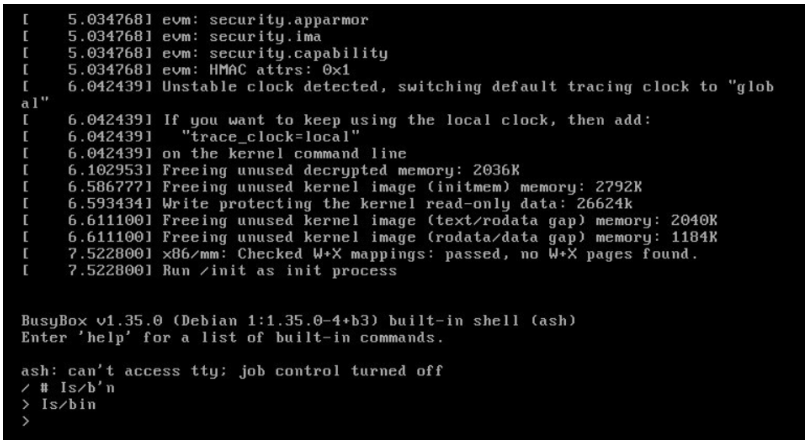
Dağıtımın Test Edilmesi

Hazırlanan **$HOME/distro/distro.iso** dağıtımımız qemu veya virtualbox ile test edilebilir.

Aşağıdaki komutla qemu ile isomuzu çalıştırıp test edebiliriz. qemu ile ilgili bilgi için ek konular bölümünden erişebilirsiniz.

```
qemu-system-x86_64 -cdrom $HOME/distro/distro.iso -m 1G
```

Eğer hatasız yapılmışsa sistem açılacak ve tty açacaktır. Birçok komutu çalışan bir dağıtım oluşturmuş olduk. Ekran görüntüsü aşağıda görülmektedir.



Dokümanın devamında kendi sistemimizi hazırlarken bu bölümde anlatılan yapıya benzer adımları yapacağız. Minimal sistemden farkı **busybox** ile oluşturduğumuz ikili dosyalarımızı(temel komutlar) kendimiz derleyeceğiz. Minimal sistemde static busybox kullandık.

Static dosyalarda dosyanın çalışması için kütüphanelerin hepsi kendi içerisine gömümülü(dahil) bir şekilde gelir. Avantajı hiçbir kütüphaneye ihtiyaç duymaz. Devevantajı ise boyutları yüksek olur. İstisnalar olsada genel olarak static tercih edilmemektedir. Static olduğunda iso boyutlarımız olması gerekenden 5-10 kat fazla olabilmektedir.

Static Minimal Dağıtım Oluşturma

Busybox ile bir dağıtım oluşturma işlemini yaptığınızı varsayıyorum. Bu aşamaya kadar başarılı bir şekilde yaptığınızı varsayarak aklınıza bir çok soru gelecektir. Bu sorulardan birini ben sorayım sizin yerinize. Busybox yoksa elimizde ya da olmasını istemiyorum nasıl olacak dağıtım diyebilirsiniz. Ufak değişiklikler olsada **busybox** distrosu hazırlarken yaptığımız aşamaların aynısı olacak. Bu durumda initrd.img dosyasını yeniden yazmamız gerekmektedir. Yukarıda initrd.img dosyası için aşağıdaki gibi bir init dosyası oluşturduğumuzu hatırlıyorsunuzdur.

```
#!busybox ash
PATH=/bin
/busybox mkdir /bin
/busybox --install -s /bin
/busybox ash
```

Daha sonra ise; **chomod +x init** komutu ile çalıştırılır yapılır. Ardından **find ./ |cpio -H newc -o >initrd.img** komutu ile **initrd.img** dosyasını oluşturmuştuk.

Şimdi bu işlemleri biraz değiştirip **busybox** dosyası yerine bağımsız bir init ikili dosyasını yazalım ve derleyelim. Bunun için;

```
distro/init.c
```

yapıyı oluşturmalıyız . Bunun için aşağıdaki komutlar çalıştırılır.

```
mkdir distro
cd distro
nano init.c
```

Komutlarından sonra **init.c** dosya içeriği aşağıdaki gibi olmalıdır.

```
#include<stdio.h>

int main()
{
char data[30];
while(1)
{
printf(">>");scanf("%s",data);
printf("girilen bilgi: %s\n",data);
}
return 0;
}
```

init.c dosyası sonsuz bir döngüde bilgi alıyor ve ekrana girilen bilgi diye tekrar yazdırılıyor. Şimdi ise **static** olarak derleyelim. **Static** derleme hiç bir başka dosyaya ihtiyaç duymadan çalışacağı anlamına gelmektedir.

gcc init.c -o init -static bu komutla static olarak derledik.

```
ldd ./init
özdevimli bir çalıştırılabilir değil
```

"özdevimli bir çalıştırılabilir değil" dinamik değil diyor yani static kısacası bir bağımlılığı yok demektir. Eğer bağımlılığı olsaydı bağımlı olduğu dosyalarıda konumlarına göre kopyalamamız gerekmekteydi.

Şimdi ise initrd.img dosyasını oluşturacak komutumuzu çalıştıralım. **echo "init"|cpio -H newc -o >initrd.img** bu komutla **initrd.img** dosyasını oluşturduk.

Bize sadece distro klasöründeki **initrd.img** dosyası daha sonra kullanmak üzere gerekli olacak.

Bir distro isosu için aşağıdaki gibi bir klasör yapısı elde etmemiz gerekmektedir.

```
distro/iso/boot/vmlinuz
distro/iso/boot/initrd.img
distro/iso/boot/grub/grub.cfg yapısını oluşturmalıyız.
```

şimdi sırasıyla satır satır yapıyı oluşturalım

```
mkdir iso
mkdir iso/boot
# sistemde kullandığım kerneli kopyaladım istenirde kernel derlenebilir.
cp /boot/vmlinuz* iso/boot/vmlinuz
# daha önce oluşturduğumuz **initrd.img** dosyamızı taşıyoruz.
mv ./initrd.img iso/boot/initrd.img
mkdir iso/boot/grub*
# grub.cfg dosyası oluşturulur ve içeriği aşağıdaki gibi düzenlenir ve kaydedilir.
touch iso/boot/grub/grub.cfg
```

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

Yukarıdaki üç satır **iso/boot/grub/grub.cfg** dosyasının içeri olacak şekilde ayarlanır.

grub-mkrescue iso/ -o distro.iso komutuyla iso doyamız oluşturulur.

Artık sistemi açabilen ve tty açıp bize suna bir yapı oluşturduk. Çalıştırmak için qemu kullanılabililir.

qemu-system-x86_64 -cdrom distro.iso -m 1G komutuyla çalıştırıp test edebiliriz.. Eğer hatasız yapılmışsa sistem açılacak ve **init** ikili dosyamız çalışacaktır. Bizden bilgi girmemizi ve daha sonra girdiğimiz bilgiyi ekrana yazan bir bağımsız dağıtım yapmış olduk.

Paket Derleme

Temel Paketler

Bir önceki bölümde minimal bir sistemi **busybox** yardımıyla tasarladık. Minimal sistem tasarımımızda temel bir sistemin nasıl hazırlandığını anlatmaya çalıştık. Eğer aşamaları takip ederek yaptığımızda kendisini açabilen bir sistem olduğunu görmekteyiz. Minimal sistemde kullanılan **busybox** aslında bizim işlerimiz çok kolaylaştırdı. Şimdi ise **busybox** ile yapabileceğimiz işlemleri kısaca şöyle sıralayabiliriz.

- Temel linux komutlarının(ls, cp, mkdir vs.) yerine busybox kullanabilir.
- Çeşitli sıkıştırma formatları(zip, tar, cpio vb.) yerine busybox kullanabilir.
- İnternete bağlanabililir(udhcpc).
- Dosya indirebilir(wget, curl vb.)
- Log(raporlama) tutabilir.
- Modülleri yönetebilir(kmod, modprobe,insmod vb.)

Bu bölümde **busybox** ile yaptımız işleri yapan paketleri derleyeceğiz. **busybox** yapamadığımız bazı işlemleri(ssh vb.) yapan paketlerde olacak. Tasarlayacağımız sistemde genel olarak şunlar yapılabilecek.

- Temel linux komutları kullanılacak
- Bash kabuğu ile tty ortamı olacak
- Sıkıştırma formatları kullanılabilecek
- Modüller yönetilebilecek
- initrd oluşturabilecek
- grub kurabilecek
- Sistemi kurabilecek
- Sistemi live olarak açabilecek
- İnternete bağlanılabilecek
- ssh bağlantısı ile uzaktan yönetilebilecek
- Metin düzenleyici editör olacak

Bu yapıda bir dağıtım için aşağıdaki paketlere ihtiyacımız olacak.

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- dostools	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-

Bir linux paketinin sorunsuz çalışabilmesi için bağımlı olduğu paketlerin önceden derlenmiş olması gerekir. Örneğin **bash** paketinin sorunsuz çalışabilmesi için **readline** ve **ncurses** kütüphaneleri gerekli. **readline** ve **ncurses** kütüphanelerinin çalışabilmesi içinde **glibc** kütüphanesi gerekli. Listede bulunan paketler sırasıyla nasıl derleneceği ayrı başlıklar altında anlatılacaktır.

Paketlere başlamadan önce şu paketleri kurmanızı tavsiye ederim.

- sudo apt-get install autoconf
- sudo apt-get install automake
- sudo apt-get install autotools-dev
- sudo apt-get install make
- sudo apt-get install meson
- sudo apt-get install cmake
- sudo apt-get install ninja-build
- sudo apt-get install pkgconf
- sudo apt install patch
- sudo apt install libtool
- sudo apt install grub-pc grub-pc-bin

Paketlerin derlenmesini **root** yetkisiyle yapınız.

base-file

Linux sistemimiz için temel ayarlamalar, dosya ve dizin yapıları olması gerekmektedir. Bu yapıyı oluşturduktan sonra sistemi bu yapının üzerine inşaa edeceğiz. Aslında linux sisteminde temel paket **glibc** paketidir. **glibc** paketinin derlenip yüklenmesinden önce temel yapının oluşturulması gerektiği için **base-file** paketi oluşturduk.

base-file Komutları

```
# Sistemin oluşturulacağı dizin yoksa oluşturuluyor
mkdir -p   $HOME/distro/rootfs
# Derleme dizini yoksa oluşturuluyor
mkdir -p  /tmp/build
# içeriği temizleniyor
rm -rf  /tmp/build/*
# Ek dosyalar kopyalanıyor. Ek dosyalar aşağıda verilmiştir.
cp -prfv files/*  /tmp/build/
# derleme konumuna geçiyoruz
cd  /tmp/build

# sistemin genel dizin yapısı oluşturuluyor
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\" >>  /tmp/build/etc/shell"
bash -c "echo 'tmpfs /tmp tmpfs rw,nodev,nosuid 0 0' >>  /tmp/build/etc/fstab"
bash -c "echo '127.0.0.1 kly' >>  /tmp/build/etc/hosts"
bash -c "echo 'kly' >  /tmp/build/etc/hostname"
bash -c "echo 'nameserver 8.8.8.8' >  /tmp/build/etc/resolv.conf"
echo root:x:0:0:root:/root:/bin/sh >  /tmp/build/etc/passwd
chmod 755  /tmp/build/etc/passwd

# tasarladığımız sistemin konumuna kopyalıyoruz.
cp -prfv  /tmp/build/*   $HOME/distro/rootfs/
```

Yukarıdaki kodları standart bir yapıya dönüştürüp aşağıdaki şablon scriptini kullanacağız.

Şablon Script Yapısı

```
#!/usr/bin/env bash
version=""
name=""
depends=""
source=""

# Paketin yükleneceği tasarlanan sistem konumu
DESTDIR="$HOME/distro/rootfs"
# Derleme konumu
ROOTBUILDDIR="/tmp/kly/build"
# Derleme yapılan paketin derleme konumun
BUILDDIR="/tmp/kly/build/build-${name}-${version}"
# paketin derleme talimatının verildiği konum
PACKAGEDIR=$(pwd)
# Paketin kaynak kodlarının olduğu konum
SOURCEDIR="/tmp/kly/build/${name}-${version}"

# derleme dizini yoksa oluşturuluyor
mkdir -p  $ROOTBUILDDIR
# içeriği temizleniyor
rm -rf $ROOTBUILDDIR/*
# dizinine geçiyoruz
cd $ROOTBUILDDIR
# Kaynak Dosya indiriliyor ve paket ismiyle açılıyor
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
# derleme dizini, yüklenecek konum dizini açılıyor ve derleme dizinine geçiliyor
mkdir -p $BUILDDIR&&mkdir -p $DESTDIR&&cd $BUILDDIR
# Paket derleme öncesi hazırlık
# ...
# Paket derlenmesi
# ...
# paket derleme sonrası yükleme ve ayarlar
# ...
```

Şablon içinde kullanılan bazı sabit bilgiler var. Bular;

- ROOTBUILDDIR="/tmp/kly/build": Derleme konumu.
- BUILDDIR="/tmp/kly/build/build-${name}-${version}": Derlenen paketin derleme konumu.
- DESTDIR="$HOME/distro/rootfs": Derlennen paketin yükleneceği(tsarladığımız sistem) konum.
- PACKAGEDIR=$(pwd) : Derleme talimatının bulunduğu(build dosyası) konum.
- SOURCEDIR="/tmp/kly/build/${name}-${version}": Derlenen paketin kaynak kodlarının konumu.

Derleme konumunu uzun uzun yazmak yerine sadece $ROOTBUILDDIR ifadesi kullanılıyor. Aslında bu işleme takma ad(alias) denir. Mesela kaynak kodların olduğu konumda bir şeyler yapmak istersek $SOURCEDIR ifadesinin kullanmamız yeterli olacaktır. Bu takma adlar tüm paketlerde geçerli olacak ifadelerdir.

Şablon Script(base-file)

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

# Paketin yükleneceği tasarlanan sistem konumu
DESTDIR="$HOME/distro/rootfs"
# Derleme konumu
ROOTBUILDDIR="/tmp/kly/build"
# Derleme yapılan paketin derleme konumun
BUILDDIR="/tmp/kly/build/build-${name}-${version}"
# paketin derleme talimatının verildiği konum
PACKAGEDIR=$(pwd)
# Paketin kaynak kodlarının olduğu konum
SOURCEDIR="/tmp/kly/build/${name}-${version}"


# initsetup
mkdir -p  $ROOTBUILDDIR #derleme dizini yoksa oluşturuluyor
rm -rf $ROOTBUILDDIR/* #içeriği temizleniyor
cd $ROOTBUILDDIR #dizinine geçiyoruz
mkdir -p $BUILDDIR&&mkdir -p $DESTDIR&&cd $BUILDDIR

# setup
cp -prfv $PACKAGEDIR/files/* $BUILDDIR/
# build

# 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 -
bash -c "echo -e \"/bin/sh \n/bin/bash \n/bin/rbash \n/bin/dash\" >> $BUILDDIR/etc/shell"
bash -c "echo 'tmpfs /tmp tmpfs rw,nodev,nosuid 0 0' >> $BUILDDIR/etc/fstab"
bash -c "echo '127.0.0.1 kly' >> $BUILDDIR/etc/hosts"
bash -c "echo 'kly' > $BUILDDIR/etc/hostname"
bash -c "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/
```

Yukarıdaki kodların sorunsuz çalışabilmesi için ek dosyayalara ihtiyaç vardır. Bu ek dosyaları indirmek için tıklayınız.

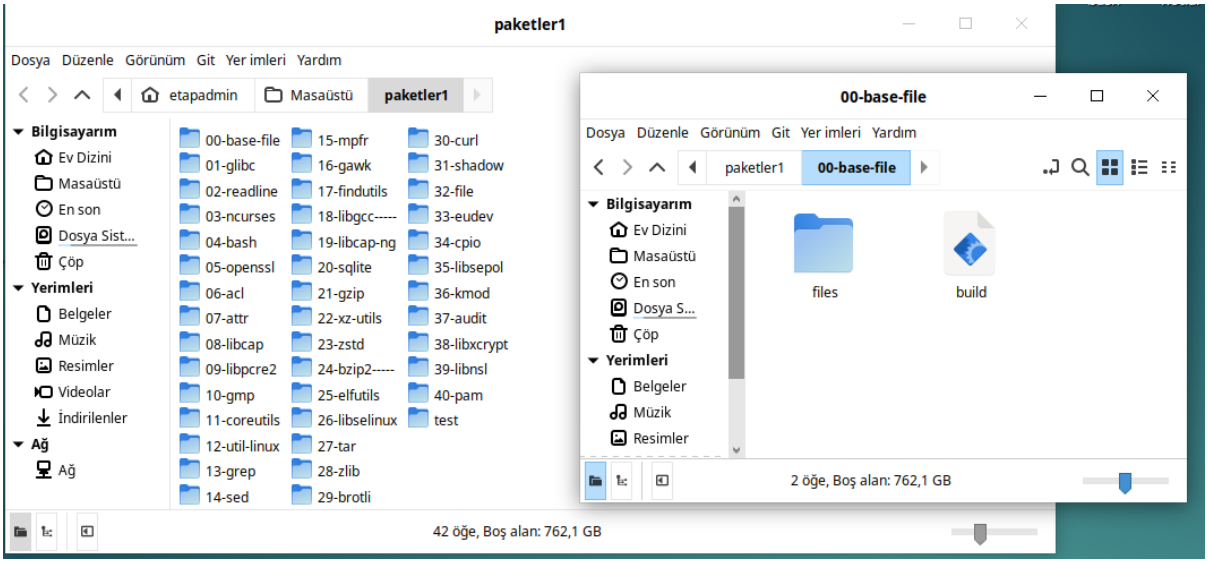
tar dosyasını indirdikten sonra istediğiniz bir konumda **base-file** adında bir dizin oluşturun ve tar dosyasını oluşturulan dizin içinde açınınız.

Yukarı verilen script kodlarını **build** adında bir dosya oluşturup içine kopyalayın ve kaydedin. Daha sonra **build** scriptini çalıştırın. Nasıl çalıştırılacağı aşağıdaki komutlarla gösterilmiştir. Aşağıda gösterilen komutları **base-file** dizinin içinde terminal açarak çalıştırınız.

```
chmod 755 build
fakeroot ./build
```

Paket Derleme Yöntemi

base-file paketleri ilk paketler olmasından dolayı detaylıca anlatıldı. Bu paketten sonraki paketlerde **şablon script** dosyası yapısında verilecektir. Script dosya altında ise ek dosyalar varsa **files.tar** şeklinde link olacaktır. Her paket için istediğiniz bir konumda bir dizin oluşturunuz. **files.tar** dosyasını oluşturulan dizin içinde açınız. Test amaçlı derleme yaptığım paketler ve **base-file** için yaptığımız dizin yapısı aşağıda gösterilmiştir.



Derleme scripti için **build** dosyası oluşturup içine yapıştırın ve kaydedin. **build** dosyasının bulunduğu dizininde terminali açarak aşağıdaki gibi çalıştırınız.

```
chmod 755 build
fakeroot ./build
```

glibc

glibc linux dağıtımlarında bütün uygulamaların çalışmasını sağlayan en temel C kütüphanesidir. **glibc** dışında diğer C standart kütüphaneler şunlardır: Bionic libc, dietlibc, EGLIBC, klibc, musl, Newlib ve uClibc. **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"
depends=""
description="temel kütüphane"
source="https://ftp.gnu.org/gnu/libc/${name}-${version}.tar.gz"
groups="sys.base"
export CC="gcc"; export CXX="g++"
# Paketin yükleneceği tasarlanan sistem konumu
DESTDIR="$HOME/distro/rootfs"
# Derleme konumu
ROOTBUILDDIR="/tmp/kly/build"
# Derleme yapılan paketin derleme konumun
BUILDDIR="/tmp/kly/build/build-${name}-${version}"
# paketin derleme talimatının verildiği konum
PACKAGEDIR=$(pwd)
# Paketin kaynak kodlarının olduğu konum
SOURCEDIR="/tmp/kly/build/${name}-${version}"

# 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 -j5 #-C $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}/etc/sysconf.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        # remove ld.so.cache file
install $BUILDDIR/files/locale-gen ${DESTDIR}/bin/locale-gen
# ek araçlar scriptleri yükleniyor
install $BUILDDIR/files/revdep-rebuild ${DESTDIR}/bin/revdep-rebuild
# dil ayarları yükleniyor
install $BUILDDIR/files/tr_TR ${DESTDIR}/usr/share/i18n/locales/tr_TR
# ldd shebang düzeltmesi yapılıyor
sed -i "s|#!/bin/bash|#!/bin/sh|g" ${DESTDIR}/usr/bin/ldd
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)
```

Bu paketin ek dosyalarını indirmek için tıklayınız.. tar dosyasını indirdikten sonra **glibc** adında bir dizin oluşturun ve tar dosyasını oluşturulan dizin içinde açınınız. Yukarı verilen script kodlarını **glibc** dizini altında **build** adında bir dosya oluşturup içine kopyalayın ve kaydedin. Oluşturulan **build** dosyasını **glibc** dizinin içinde terminal açarak çalıştırınız.

```
chmod 755 build
fakeroot ./build
```

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.

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

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.

readline

libreadline, Linux işletim sistemi için geliştirilmiş bir kütüphanedir. Bu kütüphane, kullanıcıların komut satırında girdi almasını ve düzenlemesini sağlar. Bir programcı olarak, libreadline'i kullanarak kullanıcı girdilerini okuyabilir, düzenleyebilir ve işleyebilirsiniz.

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

Derleme

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"
groups="sys.apps"

# Paketin yükleneceği tasarlanan sistem konumu
DESTDIR="$HOME/distro/rootfs"
# Derleme konumu
ROOTBUILDDIR="/tmp/kly/build"
# Derleme yapılan paketin derleme konumun
BUILDDIR="/tmp/kly/build/build-${name}-${version}"
# paketin derleme talimatının verildiği konum
PACKAGEDIR=$(pwd)
# Paketin kaynak kodlarının olduğu konum
SOURCEDIR="/tmp/kly/build/${name}-${version}"

# 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 $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
```

Bu paketin ek dosyalarını indirmek için tıklayınız. tar dosyasını indirdikten sonra **readline** adında bir dizin oluşturun ve tar dosyasını oluşturulan dizin içinde açınınız. Yukarı verilen script kodlarını **readline** dizini altında **build** adında bir dosya oluşturup içine kopyalayın ve kaydedin. Oluşturulan **build** dosyasını **readline** dizinin içinde terminal açarak çalıştırınız.

```
chmod 755 build
fakeroot ./build
```

Program Yazma

Altta görülen **readline** kütüphanesini kullanarak terminalde kullanıcıdan mesaj alan ve mesajı ekrana yazan programı hazırladık. $HOME(ev dizinimiz) dizinine merhaba.c dosyası oluşturup aşağıdaki kodları ekleyelim.

```
# merhaba.c doayası
#include<stdio.h>
#include<readline/readline.h>
void main()
{
char* msg=readline("Adını Yaz:");
puts(msg);
}
```

Program Derleme

```
cd $HOME
gcc -o merhaba merhaba.c -lreadline
cp merhaba $HOME/distro/rootfs/merhaba
```

Program Test Etme

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

Program hatasız çalışıyorsa **readline** kütüphanemiz hatasız derlenmiş olacaktır.

ncurses

ncurses, Linux işletim sistemi için bir programlama kütüphanesidir. Bu kütüphane, terminal tabanlı kullanıcı arayüzleri oluşturmak için kullanılır. ncurses, terminal ekranını kontrol etmek, metin tabanlı menüler oluşturmak, renkleri ve stil özelliklerini ayarlamak gibi işlevlere sahiptir.

ncurses, kullanıcıya metin tabanlı bir arayüz sağlar ve terminal penceresinde çeşitli işlemler gerçekleştirmek için kullanılabilir. Örneğin, bir metin düzenleyici, dosya tarayıcısı veya metin tabanlı bir oyun gibi uygulamalar ncurses kullanarak geliştirilebilir.

Derleme

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"
groups="sys.libs"

# Paketin yükleneceği tasarlanan sistem konumu
DESTDIR="$HOME/distro/rootfs"
# Derleme konumu
ROOTBUILDDIR="/tmp/kly/build"
# Derleme yapılan paketin derleme konumun
BUILDDIR="/tmp/kly/build/build-${name}-${version}"
# paketin derleme talimatının verildiği konum
PACKAGEDIR=$(pwd)
# Paketin kaynak kodlarının olduğu konum
SOURCEDIR="/tmp/kly/build/${name}-${version}"

# 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
./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 -j5 #-C $DESTDIR all

# 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

# make sure that anything linking against it links against libncurses.so instead
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/${lib}.pc"
        fi
done
# make sure that anything linking against it links against libncursesw.so instead
for lib in tic tinfo tinfow ticw; do
        if [ ! -f "$DESTDIR/usr/lib64/pkgconfig/${lib}.pc" ]; then
                ln -sv ncursesw.pc "$DESTDIR/usr/lib64/pkgconfig/${lib}.pc"
        fi
done
# legacy binary support
for lib in libncursesw libncurses libtinfo libpanelw libformw libmenuw ; do
        ln -sv ${lib}.so.${so_ver} ${lib}.so.5
done
```

Paket adında(ncurses) istediğiniz bir konumda bir dizin oluşturun ve dizin içine giriniz. Yukarı verilen script kodlarını build adında bir dosya oluşturup içine kopyalayın ve kaydedin. Daha sonra build scriptini çalıştırın. Nasıl çalıştırılacağı aşağıdaki komutlarla gösterilmiştir. Aşağıda gösterilen komutları paket için oluşturulan dizinin içinde terminal açarak çalıştırınız.

```
chmod 755 build
fakeroot ./build
```

bash

Bash, Linux ve diğer Unix tabanlı işletim sistemlerinde kullanılan bir kabuk programlama dilidir. Kullanıcıların komutlar vererek işletim sistemini yönetmelerine olanak tanır. Bash, kullanıcıların işlemleri otomatikleştirmesine ve betik dosyaları oluşturmasına olanak tanır. Özellikle sistem yöneticileri ve geliştiriciler arasında yaygın olarak kullanılan güçlü bir araçtır.

Derleme

```
#!/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"
groups="app.shell"

# Paketin yükleneceği tasarlanan sistem konumu
DESTDIR="$HOME/distro/rootfs"
# Derleme konumu
ROOTBUILDDIR="/tmp/kly/build"
# Derleme yapılan paketin derleme konumun
BUILDDIR="/tmp/kly/build/build-${name}-${version}"
# paketin derleme talimatının verildiği konum
PACKAGEDIR=$(pwd)
# Paketin kaynak kodlarının olduğu konum
SOURCEDIR="/tmp/kly/build/${name}-${version}"

# 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 $BUILDDIR

# setup
./configure --prefix=/usr --libdir=/usr/lib64 --bindir=/bin \
        --with-curses --enable-readline --without-bash-malloc
# build
make -j5 #-C $DESTDIR all

# package
make install DESTDIR=$DESTDIR
cd $DESTDIR/bin
ln -s bash sh
```

Paket adında(bash) istediğiniz bir konumda bir dizin oluşturun ve dizin içine giriniz. Yukarı verilen script kodlarını build adında bir dosya oluşturup içine kopyalayın ve kaydedin. Daha sonra build scriptini çalıştırın. Nasıl çalıştırılacağı aşağıdaki komutlarla gösterilmiştir. Aşağıda gösterilen komutları paket için oluşturulan dizinin içinde terminal açarak çalıştırınız.

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

openssl

OpenSSL, açık kaynaklı bir kriptografik kütüphanedir. 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"
source="https://www.openssl.org/source/${name}-${version}.tar.gz"
# Paketin yükleneceği tasarlanan sistem konumu
DESTDIR="$HOME/distro/rootfs"
# Derleme konumu
ROOTBUILDDIR="/tmp/kly/build"
# Derleme yapılan paketin derleme konumun
BUILDDIR="/tmp/kly/build/build-${name}-${version}"
# paketin derleme talimatının verildiği konum
PACKAGEDIR=$(pwd)
# Paketin kaynak kodlarının olduğu konum
SOURCEDIR="/tmp/kly/build/${name}-${version}"

# 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 $BUILDDIR

# 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 -j5 #-C $DESTDIR all

# 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  $jobs
```

Bu paketin ek dosyalarını indirmek için tıklayınız.. tar dosyasını indirdikten sonra **openssl** adında bir dizin oluşturun ve tar dosyasını oluşturulan dizin içinde açınınız. Yukarı verilen script kodlarını **openssl** dizini altında **build** adında bir dosya oluşturup içine kopyalayın ve kaydedin. Oluşturulan **build** dosyasını **openssl** dizinin içinde terminal açarak çalıştırınız.

```
chmod 755 build
fakeroot ./build
```

acl

ACL (Access Control List), dosya sistemlerinde veya ağ cihazlarında erişim kontrolünü yönetmek için kullanılan bir mekanizmadır. ACL'ler, belirli kullanıcıların veya kullanıcı gruplarının belirli dosya veya kaynaklara erişim düzeylerini tanımlamak için kullanılır. Bu, dosyalara veya kaynaklara erişim izinlerini hassas bir şekilde kontrol etmeyi sağlar. Örneğin, bir dosyaya sadece belirli kullanıcıların yazma izni vermek için ACL'ler kullanılabilir. Bu şekilde, güvenlik ve erişim kontrolü daha ayrıntılı bir şekilde yapılabilmektedir.coreutils için gerekli olan paket.

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"
source="https://download.savannah.nongnu.org/releases/acl/acl-$version.tar.xz"
depends="attr"

# Paketin yükleneceği tasarlanan sistem konumu
DESTDIR="$HOME/distro/rootfs"
# Derleme konumu
ROOTBUILDDIR="/tmp/kly/build"
# Derleme yapılan paketin derleme konumun
BUILDDIR="/tmp/kly/build/build-${name}-${version}"
# paketin derleme talimatının verildiği konum
PACKAGEDIR=$(pwd)
# Paketin kaynak kodlarının olduğu konum
SOURCEDIR="/tmp/kly/build/${name}-${version}"

# 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 $BUILDDIR

# setup
./configure --prefix=/usr --libdir=/usr/lib64

# build
make -j5 #-C $DESTDIR all
# package
make install DESTDIR=$DESTDIR
```

Paket adında(acl) istediğiniz bir konumda bir dizin oluşturun ve dizin içine giriniz. Yukarı verilen script kodlarını build adında bir dosya oluşturup içine kopyalayın ve kaydedin. Daha sonra build scriptini çalıştırın. Nasıl çalıştırılacağı aşağıdaki komutlarla gösterilmiştir. Aşağıda gösterilen komutları paket için oluşturulan dizinin içinde terminal açarak çalıştırınız.

```
chmod 755 build
fakeroot ./build
```

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. Örneğin, bir dosyanın izinlerini değiştirmek veya bir dosyanın sahibini görmek için attr komutunu kullanabilirsiniz.

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=""
group="sys.apps"


# Paketin yükleneceği tasarlanan sistem konumu
DESTDIR="$HOME/distro/rootfs"
# Derleme konumu
ROOTBUILDDIR="/tmp/kly/build"
# Derleme yapılan paketin derleme konumun
BUILDDIR="/tmp/kly/build/build-${name}-${version}"
# paketin derleme talimatının verildiği konum
PACKAGEDIR=$(pwd)
# Paketin kaynak kodlarının olduğu konum
SOURCEDIR="/tmp/kly/build/${name}-${version}"

# 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 $BUILDDIR

# setup
./configure --prefix=/usr --sysconfdir=/etc --libdir=/usr/lib64

# build
make

# package
make install DESTDIR=$DESTDIR
```

Paket adında(attr) istediğiniz bir konumda bir dizin oluşturun ve dizin içine giriniz. Yukarı verilen script kodlarını build adında bir dosya oluşturup içine kopyalayın ve kaydedin. Daha sonra build scriptini çalıştırın. Nasıl çalıştırılacağı aşağıdaki komutlarla gösterilmiştir. Aşağıda gösterilen komutları paket için oluşturulan dizinin içinde terminal açarak çalıştırınız.

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

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"
groups="net.misc"

# Paketin yükleneceği tasarlanan sistem konumu
DESTDIR="$HOME/distro/rootfs"
# Derleme konumu
ROOTBUILDDIR="/tmp/kly/build"
# Derleme yapılan paketin derleme konumun
BUILDDIR="/tmp/kly/build/build-${name}-${version}"
# paketin derleme talimatının verildiği konum
PACKAGEDIR=$(pwd)
# Paketin kaynak kodlarının olduğu konum
SOURCEDIR="/tmp/kly/build/${name}-${version}"

# 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
_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'"
)
# 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
```

Paket adında(libcap) istediğiniz bir konumda bir dizin oluşturun ve dizin içine giriniz. Yukarı verilen script kodlarını build adında bir dosya oluşturup içine kopyalayın ve kaydedin. Daha sonra build scriptini çalıştırın. Nasıl çalıştırılacağı aşağıdaki komutlarla gösterilmiştir. Aşağıda gösterilen komutları paket için oluşturulan dizinin içinde terminal açarak çalıştırınız.

```
chmod 755 build
fakeroot ./build
```

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"
group="dev.libs"

# Paketin yükleneceği tasarlanan sistem konumu
DESTDIR="$HOME/distro/rootfs"
# Derleme konumu
ROOTBUILDDIR="/tmp/kly/build"
# Derleme yapılan paketin derleme konumun
BUILDDIR="/tmp/kly/build/build-${name}-${version}"
# paketin derleme talimatının verildiği konum
PACKAGEDIR=$(pwd)
# Paketin kaynak kodlarının olduğu konum
SOURCEDIR="/tmp/kly/build/${name}-${version}"

# 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
./configure --prefix=/usr --enable-shared --enable-static \
        --enable-pcre2-16 --enable-pcre2-32 --enable-jit --enable-pcre2test-libreadline
# build
make -j5 #-C $DESTDIR all

# package
make install DESTDIR=$DESTDIR
```

Paket adında(libpcre2) istediğiniz bir konumda bir dizin oluşturun ve dizin içine giriniz. Yukarı verilen script kodlarını build adında bir dosya oluşturup içine kopyalayın ve kaydedin. Daha sonra build scriptini çalıştırın. Nasıl çalıştırılacağı aşağıdaki komutlarla gösterilmiştir. Aşağıda gösterilen komutları paket için oluşturulan dizinin içinde terminal açarak çalıştırınız.

```
chmod 755 build
fakeroot ./build
```

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"
groups="dev.libs"

# Paketin yükleneceği tasarlanan sistem konumu
DESTDIR="$HOME/distro/rootfs"
# Derleme konumu
ROOTBUILDDIR="/tmp/kly/build"
# Derleme yapılan paketin derleme konumun
BUILDDIR="/tmp/kly/build/build-${name}-${version}"
# paketin derleme talimatının verildiği konum
PACKAGEDIR=$(pwd)
# Paketin kaynak kodlarının olduğu konum
SOURCEDIR="/tmp/kly/build/${name}-${version}"

# initsetup
# derleme dizini yoksa oluşturuluyor
mkdir -p  $ROOTBUILDDIR
# içeriği temizleniyor
rm -rf $ROOTBUILDDIR/*
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 $BUILDDIR

# setup
./configure --prefix=/usr --libdir=/usr/lib64 cxx --enable-cxx --enable-fat

# build
make -j5 #-C $DESTDIR all

# package
make install DESTDIR=${DESTDIR}
```

Paket adında(gmp) istediğiniz bir konumda bir dizin oluşturun ve dizin içine giriniz. Yukarı verilen script kodlarını build adında bir dosya oluşturup içine kopyalayın ve kaydedin. Daha sonra build scriptini çalıştırın. Nasıl çalıştırılacağı aşağıdaki komutlarla gösterilmiştir. Aşağıda gösterilen komutları paket için oluşturulan dizinin içinde terminal açarak çalıştırınız.

```
chmod 755 build
fakeroot ./build
```

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="9.5"
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"
group="sys.apps"

# Paketin yükleneceği tasarlanan sistem konumu
DESTDIR="$HOME/distro/rootfs"
# Derleme konumu
ROOTBUILDDIR="/tmp/kly/build"
# Derleme yapılan paketin derleme konumun
BUILDDIR="/tmp/kly/build/build-${name}-${version}"
# paketin derleme talimatının verildiği konum
PACKAGEDIR=$(pwd)
# Paketin kaynak kodlarının olduğu konum
SOURCEDIR="/tmp/kly/build/${name}-${version}"

# initsetup
# derleme dizini yoksa oluşturuluyor
mkdir -p  $ROOTBUILDDIR
# içeriği temizleniyor
rm -rf $ROOTBUILDDIR/*
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 $BUILDDIR

# setup
export CFLAGS="-static-libgcc -static-libstdc++ -fPIC"
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 -j5 #-C $DESTDIR all

# package
make install DESTDIR=$DESTDIR
```

Paket adında(coreutils) istediğiniz bir konumda bir dizin oluşturun ve dizin içine giriniz. Yukarı verilen script kodlarını build adında bir dosya oluşturup içine kopyalayın ve kaydedin. Daha sonra build scriptini çalıştırın. Nasıl çalıştırılacağı aşağıdaki komutlarla gösterilmiştir. Aşağıda gösterilen komutları paket için oluşturulan dizinin içinde terminal açarak çalıştırınız.

```
chmod 755 build
fakeroot ./build
```

util-linux

Util-linux paketi, Linux tabanlı sistemlerde kritik öneme sahip bir bileşendir. Bu paket, dosya sistemleri, disk yönetimi, kullanıcı yönetimi ve sistem izleme gibi çeşitli işlevleri yerine getiren bir dizi araç içerir. Örneğin, mount ve umount komutları, dosya sistemlerini bağlamak ve ayırmak için kullanılırken, fdisk ve parted disk bölümlerini yönetmek için kullanılır. Ayrıca, login, su, ve passwd gibi komutlar, kullanıcı kimlik doğrulama ve yönetimi için önemli işlevler sunar.

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"
group="sys.apps"

# Paketin yükleneceği tasarlanan sistem konumu
DESTDIR="$HOME/distro/rootfs"
# Derleme konumu
ROOTBUILDDIR="/tmp/kly/build"
# Derleme yapılan paketin derleme konumun
BUILDDIR="/tmp/kly/build/build-${name}-${version}"
# paketin derleme talimatının verildiği konum
PACKAGEDIR=$(pwd)
# Paketin kaynak kodlarının olduğu konum
SOURCEDIR="/tmp/kly/build/${name}-${version}"

# initsetup
# derleme dizini yoksa oluşturuluyor
mkdir -p  $ROOTBUILDDIR
# içeriği temizleniyor
rm -rf $ROOTBUILDDIR/*
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 $BUILDDIR

# 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 -j5 #-C $DESTDIR all

# package
make install DESTDIR=$DESTDIR
```

Yukarıdaki kodların sorunsuz çalışabilmesi için ek dosyayalara ihtiyaç vardır. Bu ek dosyaları indirmek için tıklayınız.

tar dosyasını indirdikten sonra istediğiniz bir konumda **util-linux** adında bir dizin oluşturun ve tar dosyasını oluşturulan dizin içinde açınınız.

Paket adında(util-linux) istediğiniz bir konumda bir dizin oluşturun ve dizin içine giriniz. Yukarı verilen script kodlarını build adında bir dosya oluşturup içine kopyalayın ve kaydedin. Daha sonra build scriptini çalıştırın. Nasıl çalıştırılacağı aşağıdaki komutlarla gösterilmiştir. Aşağıda gösterilen komutları paket için oluşturulan dizinin içinde terminal açarak çalıştırınız.

```
chmod 755 build
fakeroot ./build
```

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"
group="sys.apps"


# Paketin yükleneceği tasarlanan sistem konumu
DESTDIR="$HOME/distro/rootfs"
# Derleme konumu
ROOTBUILDDIR="/tmp/kly/build"
# Derleme yapılan paketin derleme konumun
BUILDDIR="/tmp/kly/build/build-${name}-${version}"
# paketin derleme talimatının verildiği konum
PACKAGEDIR=$(pwd)
# Paketin kaynak kodlarının olduğu konum
SOURCEDIR="/tmp/kly/build/${name}-${version}"

# initsetup
# derleme dizini yoksa oluşturuluyor
mkdir -p  $ROOTBUILDDIR
# içeriği temizleniyor
rm -rf $ROOTBUILDDIR/*
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 $BUILDDIR

# setup
./configure --prefix=/usr --libdir=/usr/lib64/

# build
make

# package
make install DESTDIR=$DESTDIR
```

Paket adında(grep) istediğiniz bir konumda bir dizin oluşturun ve dizin içine giriniz. Yukarı verilen script kodlarını build adında bir dosya oluşturup içine kopyalayın ve kaydedin. Daha sonra build scriptini çalıştırın. Nasıl çalıştırılacağı aşağıdaki komutlarla gösterilmiştir. Aşağıda gösterilen komutları paket için oluşturulan dizinin içinde terminal açarak çalıştırınız.

```
chmod 755 build
fakeroot ./build
```

sed

sed, "stream editor" anlamına gelen bir Linux komut satırı aracıdır. Temel görevi, metin akışını düzenlemek ve dönüştürmektir. 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"
group="sys.apps"

# Paketin yükleneceği tasarlanan sistem konumu
DESTDIR="$HOME/distro/rootfs"
# Derleme konumu
ROOTBUILDDIR="/tmp/kly/build"
# Derleme yapılan paketin derleme konumun
BUILDDIR="/tmp/kly/build/build-${name}-${version}"
# paketin derleme talimatının verildiği konum
PACKAGEDIR=$(pwd)
# Paketin kaynak kodlarının olduğu konum
SOURCEDIR="/tmp/kly/build/${name}-${version}"

# initsetup
# derleme dizini yoksa oluşturuluyor
mkdir -p  $ROOTBUILDDIR
# içeriği temizleniyor
rm -rf $ROOTBUILDDIR/*
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 $BUILDDIR

# setup
./configure --prefix=/usr --libdir=/usr/lib64/

# build
make

# package
make install DESTDIR=$DESTDIR
```

Paket adında(sed) istediğiniz bir konumda bir dizin oluşturun ve dizin içine giriniz. Yukarı verilen script kodlarını build adında bir dosya oluşturup içine kopyalayın ve kaydedin. Daha sonra build scriptini çalıştırın. Nasıl çalıştırılacağı aşağıdaki komutlarla gösterilmiştir. Aşağıda gösterilen komutları paket için oluşturulan dizinin içinde terminal açarak çalıştırınız.

```
chmod 755 build
fakeroot ./build
```

mpfr

MPFR (Multiple Precision Floating-Point Reliable) paketi, yüksek hassasiyetli kayan nokta hesaplamaları yapmak için kullanılan bir kütüphanedir. Bu kütüphane, matematiksel hesaplamalarda doğruluğu artırmak amacıyla tasarlanmıştır.

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"
groups="dev.libs"

# Paketin yükleneceği tasarlanan sistem konumu
DESTDIR="$HOME/distro/rootfs"
# Derleme konumu
ROOTBUILDDIR="/tmp/kly/build"
# Derleme yapılan paketin derleme konumun
BUILDDIR="/tmp/kly/build/build-${name}-${version}"
# paketin derleme talimatının verildiği konum
PACKAGEDIR=$(pwd)
# Paketin kaynak kodlarının olduğu konum
SOURCEDIR="/tmp/kly/build/${name}-${version}"

# initsetup
# derleme dizini yoksa oluşturuluyor
mkdir -p  $ROOTBUILDDIR
# içeriği temizleniyor
rm -rf $ROOTBUILDDIR/*
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 $BUILDDIR

# setup
./configure --prefix=/usr --libdir=/usr/lib64 --enable-shared \
--enable-static --disable-maintainer-mode --enable-thread-safe

# build
make

# package
make install DESTDIR=${DESTDIR}
```

Paket adında(mpfr) istediğiniz bir konumda bir dizin oluşturun ve dizin içine giriniz. Yukarı verilen script kodlarını build adında bir dosya oluşturup içine kopyalayın ve kaydedin. Daha sonra build scriptini çalıştırın. Nasıl çalıştırılacağı aşağıdaki komutlarla gösterilmiştir. Aşağıda gösterilen komutları paket için oluşturulan dizinin içinde terminal açarak çalıştırınız.

```
chmod 755 build
fakeroot ./build
```

gawk

gawk, GNU projesinin bir parçası olarak geliştirilmiş bir metin işleme aracıdır. "awk" dilinin GNU versiyonu olan gawk, özellikle metin dosyalarındaki verileri analiz etmek, düzenlemek ve raporlamak için kullanılır. gawk, satır ve sütun bazında veri işleme yeteneği ile kullanıcıların karmaşık metin manipülasyonları gerçekleştirmesine olanak tanır.

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"
group="sys.apps"

# Paketin yükleneceği tasarlanan sistem konumu
DESTDIR="$HOME/distro/rootfs"
# Derleme konumu
ROOTBUILDDIR="/tmp/kly/build"
# Derleme yapılan paketin derleme konumun
BUILDDIR="/tmp/kly/build/build-${name}-${version}"
# paketin derleme talimatının verildiği konum
PACKAGEDIR=$(pwd)
# Paketin kaynak kodlarının olduğu konum
SOURCEDIR="/tmp/kly/build/${name}-${version}"

# initsetup
# derleme dizini yoksa oluşturuluyor
mkdir -p  $ROOTBUILDDIR
# içeriği temizleniyor
rm -rf $ROOTBUILDDIR/*
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 $BUILDDIR

# setup
./configure --prefix=/usr --libdir=/usr/lib64/ --sysconfdir=/etc --without-libsigsegv

# build
make

# package
make install DESTDIR=$DESTDIR
```

Paket adında(gawk) istediğiniz bir konumda bir dizin oluşturun ve dizin içine giriniz. Yukarı verilen script kodlarını build adında bir dosya oluşturup içine kopyalayın ve kaydedin. Daha sonra build scriptini çalıştırın. Nasıl çalıştırılacağı aşağıdaki komutlarla gösterilmiştir. Aşağıda gösterilen komutları paket için oluşturulan dizinin içinde terminal açarak çalıştırınız.

```
chmod 755 build
fakeroot ./build
```

findutils

Finutils paketi, Linux işletim sistemlerinde dosya ve dizin yönetimi ile ilgili çeşitli işlevler sunan bir araç setidir. Bu paket, özellikle dosya sistemleri üzerinde işlem yaparken kullanıcıların işini kolaylaştırmayı hedefler.

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=""
group="sys.apps"

# Paketin yükleneceği tasarlanan sistem konumu
DESTDIR="$HOME/distro/rootfs"
# Derleme konumu
ROOTBUILDDIR="/tmp/kly/build"
# Derleme yapılan paketin derleme konumun
BUILDDIR="/tmp/kly/build/build-${name}-${version}"
# paketin derleme talimatının verildiği konum
PACKAGEDIR=$(pwd)
# Paketin kaynak kodlarının olduğu konum
SOURCEDIR="/tmp/kly/build/${name}-${version}"

# initsetup
# derleme dizini yoksa oluşturuluyor
mkdir -p  $ROOTBUILDDIR
# içeriği temizleniyor
rm -rf $ROOTBUILDDIR/*
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 $BUILDDIR

# setup
./configure --prefix=/usr --libdir=/usr/lib64/

# build
make

# package
make install DESTDIR=$DESTDIR
```

Paket adında(findutils) istediğiniz bir konumda bir dizin oluşturun ve dizin içine giriniz. Yukarı verilen script kodlarını build adında bir dosya oluşturup içine kopyalayın ve kaydedin. Daha sonra build scriptini çalıştırın. Nasıl çalıştırılacağı aşağıdaki komutlarla gösterilmiştir. Aşağıda gösterilen komutları paket için oluşturulan dizinin içinde terminal açarak çalıştırınız.

```
chmod 755 build
fakeroot ./build
```

gcc

gcc paketi, GNU Compiler Collection (GCC) ile birlikte gelen ve C, C++ gibi dillerde yazılmış programların çalışması için gerekli olan temel kütüphaneleri içeren bir bileşendir. Bu paket, derleyici tarafından üretilen kodun çalışabilmesi için gerekli olan düşük seviyeli işlevleri sağlar. Debian ortamında derlemek için **sudo apt install libmpc-dev libmpfr-dev libgmp-dev libisl-dev** komutuyla paketleri kurmalıyız.

Derleme

```
#!/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"
groups="sys.devel"

# Paketin yükleneceği tasarlanan sistem konumu
DESTDIR="$HOME/distro/rootfs"
# Derleme konumu
ROOTBUILDDIR="/tmp/kly/build"
# Derleme yapılan paketin derleme konumun
BUILDDIR="/tmp/kly/build/build-${name}-${version}"
# paketin derleme talimatının verildiği konum
PACKAGEDIR=$(pwd)
# Paketin kaynak kodlarının olduğu konum
SOURCEDIR="/tmp/kly/build/${name}-${version}"

# initsetup
# derleme dizini yoksa oluşturuluyor
mkdir -p  $ROOTBUILDDIR
# içeriği temizleniyor
rm -rf $ROOTBUILDDIR/*
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 $BUILDDIR

# setup
export CFLAGS="-O2 -s"
export CXXFLAGS="-O2 -s"
unset LDFLAGS
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)
```

Paket adında(gcc) istediğiniz bir konumda bir dizin oluşturun ve dizin içine giriniz. Yukarı verilen script kodlarını build adında bir dosya oluşturup içine kopyalayın ve kaydedin. Daha sonra build scriptini çalıştırın. Nasıl çalıştırılacağı aşağıdaki komutlarla gösterilmiştir. Aşağıda gösterilen komutları paket için oluşturulan dizinin içinde terminal açarak çalıştırınız.

```
chmod 755 build
fakeroot ./build
```

libcap-ng

libcap-ng paketi, Linux işletim sistemlerinde yetki yönetimi ve güvenlik uygulamaları için kullanılan bir kütüphanedir. Bu kütüphane, kullanıcıların ve süreçlerin sahip olduğu yetkileri daha esnek bir şekilde yönetmelerine olanak tanır.

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"
groups="sys.libs"

# Paketin yükleneceği tasarlanan sistem konumu
DESTDIR="$HOME/distro/rootfs"
# Derleme konumu
ROOTBUILDDIR="/tmp/kly/build"
# Derleme yapılan paketin derleme konumun
BUILDDIR="/tmp/kly/build/build-${name}-${version}"
# paketin derleme talimatının verildiği konum
PACKAGEDIR=$(pwd)
# Paketin kaynak kodlarının olduğu konum
SOURCEDIR="/tmp/kly/build/${name}-${version}"

# initsetup
# derleme dizini yoksa oluşturuluyor
mkdir -p  $ROOTBUILDDIR
# içeriği temizleniyor
rm -rf $ROOTBUILDDIR/*
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 $BUILDDIR

# setup
./autogen.sh
./configure --prefix=/usr --libdir=/lib64 --with-python --with-python3

# build
make

# package
make install DESTDIR=$DESTDIR
```

Paket adında(libcap-ng) istediğiniz bir konumda bir dizin oluşturun ve dizin içine giriniz. Yukarı verilen script kodlarını build adında bir dosya oluşturup içine kopyalayın ve kaydedin. Daha sonra build scriptini çalıştırın. Nasıl çalıştırılacağı aşağıdaki komutlarla gösterilmiştir. Aşağıda gösterilen komutları paket için oluşturulan dizinin içinde terminal açarak çalıştırınız.

```
chmod 755 build
fakeroot ./build
```

sqlite

SQLite, C dilinde yazılmış ve gömülü bir veritabanı motoru olarak bilinen bir yazılımdır. Özellikle hafifliği ve taşınabilirliği ile dikkat çeker. SQLite, bir dosya sistemi üzerinde çalışarak verileri depolar ve bu sayede herhangi bir sunucuya ihtiyaç duymadan uygulama içinde kullanılabilir.

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"
group="dev.db"

# Paketin yükleneceği tasarlanan sistem konumu
DESTDIR="$HOME/distro/rootfs"
# Derleme konumu
ROOTBUILDDIR="/tmp/kly/build"
# Derleme yapılan paketin derleme konumun
BUILDDIR="/tmp/kly/build/build-${name}-${version}"
# paketin derleme talimatının verildiği konum
PACKAGEDIR=$(pwd)
# Paketin kaynak kodlarının olduğu konum
SOURCEDIR="/tmp/kly/build/${name}-${version}"

# initsetup
# derleme dizini yoksa oluşturuluyor
mkdir -p  $ROOTBUILDDIR
# içeriği temizleniyor
rm -rf $ROOTBUILDDIR/*
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 $BUILDDIR

# 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
```

Paket adında(sqlite) istediğiniz bir konumda bir dizin oluşturun ve dizin içine giriniz. Yukarı verilen script kodlarını build adında bir dosya oluşturup içine kopyalayın ve kaydedin. Daha sonra build scriptini çalıştırın. Nasıl çalıştırılacağı aşağıdaki komutlarla gösterilmiştir. Aşağıda gösterilen komutları paket için oluşturulan dizinin içinde terminal açarak çalıştırınız.

```
chmod 755 build
fakeroot ./build
```

gzip

Gzip, "GNU zip" ifadesinin kısaltmasıdır ve dosyaları sıkıştırmak için kullanılan bir yazılımdır. Temel amacı, dosya boyutunu azaltarak depolama alanından tasarruf sağlamak ve veri iletimini hızlandırmaktır. Gzip, genellikle metin dosyaları gibi tekrarlayan verilerde yüksek sıkıştırma oranları sunar.

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"
groups="app.arch"

# Paketin yükleneceği tasarlanan sistem konumu
DESTDIR="$HOME/distro/rootfs"
# Derleme konumu
ROOTBUILDDIR="/tmp/kly/build"
# Derleme yapılan paketin derleme konumun
BUILDDIR="/tmp/kly/build/build-${name}-${version}"
# paketin derleme talimatının verildiği konum
PACKAGEDIR=$(pwd)
# Paketin kaynak kodlarının olduğu konum
SOURCEDIR="/tmp/kly/build/${name}-${version}"

# initsetup
# derleme dizini yoksa oluşturuluyor
mkdir -p  $ROOTBUILDDIR
# içeriği temizleniyor
rm -rf $ROOTBUILDDIR/*
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 $BUILDDIR

# setup
export DEFS="NO_ASM"
./autoreconf -fvi
./configure --prefix=/usr --libdir=/usr/lib64/

# build
make

# package
make install DESTDIR=$DESTDIR
```

Paket adında(gzip) istediğiniz bir konumda bir dizin oluşturun ve dizin içine giriniz. Yukarı verilen script kodlarını build adında bir dosya oluşturup içine kopyalayın ve kaydedin. Daha sonra build scriptini çalıştırın. Nasıl çalıştırılacağı aşağıdaki komutlarla gösterilmiştir. Aşağıda gösterilen komutları paket için oluşturulan dizinin içinde terminal açarak çalıştırınız.

```
chmod 755 build
fakeroot ./build
```

xz-utils

xz-utils, Linux işletim sistemlerinde dosyaları sıkıştırmak ve açmak için kullanılan bir yazılım paketidir. Bu paket, LZMA (Lempel-Ziv-Markov chain algorithm) algoritmasını temel alarak yüksek sıkıştırma oranları sağlar. xz-utils, genellikle büyük dosyaların depolanması ve aktarılması sırasında disk alanından tasarruf sağlamak amacıyla tercih edilir.

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"
group="app.arch"

# Paketin yükleneceği tasarlanan sistem konumu
DESTDIR="$HOME/distro/rootfs"
# Derleme konumu
ROOTBUILDDIR="/tmp/kly/build"
# Derleme yapılan paketin derleme konumun
BUILDDIR="/tmp/kly/build/build-${name}-${version}"
# paketin derleme talimatının verildiği konum
PACKAGEDIR=$(pwd)
# Paketin kaynak kodlarının olduğu konum
SOURCEDIR="/tmp/kly/build/${name}-${version}"

# initsetup
# derleme dizini yoksa oluşturuluyor
mkdir -p  $ROOTBUILDDIR
# içeriği temizleniyor
rm -rf $ROOTBUILDDIR/*
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 $BUILDDIR

# setup
./configure --prefix=/usr --libdir=/usr/lib64 --enable-static \
--enable-shared --enable-doc --enable-nls
# build
make $jobs

# package
make install DESTDIR=$DESTDIR
```

Paket adında(xz-utils) istediğiniz bir konumda bir dizin oluşturun ve dizin içine giriniz. Yukarı verilen script kodlarını build adında bir dosya oluşturup içine kopyalayın ve kaydedin. Daha sonra build scriptini çalıştırın. Nasıl çalıştırılacağı aşağıdaki komutlarla gösterilmiştir. Aşağıda gösterilen komutları paket için oluşturulan dizinin içinde terminal açarak çalıştırınız.

```
chmod 755 build
fakeroot ./build
```

zstd

Zstd (Zstandard), Facebook tarafından geliştirilen bir veri sıkıştırma algoritmasıdır. Bu paket, verilerin boyutunu azaltarak depolama alanından tasarruf sağlamak ve veri iletimini hızlandırmak amacıyla kullanılır. Zstd, hem sıkıştırma hem de açma işlemlerinde yüksek hız ve verimlilik sunar. Özellikle büyük veri setleri ile çalışırken, Zstd'nin sunduğu sıkıştırma oranları, diğer geleneksel algoritmalara göre daha üstündür.

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"
groups="libs"

# Paketin yükleneceği tasarlanan sistem konumu
DESTDIR="$HOME/distro/rootfs"
# Derleme konumu
ROOTBUILDDIR="/tmp/kly/build"
# Derleme yapılan paketin derleme konumun
BUILDDIR="/tmp/kly/build/build-${name}-${version}"
# paketin derleme talimatının verildiği konum
PACKAGEDIR=$(pwd)
# Paketin kaynak kodlarının olduğu konum
SOURCEDIR="/tmp/kly/build/${name}-${version}"

# initsetup
# derleme dizini yoksa oluşturuluyor
mkdir -p  $ROOTBUILDDIR
# içeriği temizleniyor
rm -rf $ROOTBUILDDIR/*
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 $BUILDDIR

# setup
# build
make

# package
make prefix=/usr install DESTDIR=$DESTDIR
```

Paket adında(zstd) istediğiniz bir konumda bir dizin oluşturun ve dizin içine giriniz. Yukarı verilen script kodlarını build adında bir dosya oluşturup içine kopyalayın ve kaydedin. Daha sonra build scriptini çalıştırın. Nasıl çalıştırılacağı aşağıdaki komutlarla gösterilmiştir. Aşağıda gösterilen komutları paket için oluşturulan dizinin içinde terminal açarak çalıştırınız.

```
chmod 755 build
fakeroot ./build
```

bzip2

bzip2, dosyaları sıkıştırmak için kullanılan bir yazılımdır ve genellikle Unix tabanlı sistemlerde tercih edilmektedir.

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"
groups="app.arch"

# Paketin yükleneceği tasarlanan sistem konumu
DESTDIR="$HOME/distro/rootfs"
# Derleme konumu
ROOTBUILDDIR="/tmp/kly/build"
# Derleme yapılan paketin derleme konumun
BUILDDIR="/tmp/kly/build/build-${name}-${version}"
# paketin derleme talimatının verildiği konum
PACKAGEDIR=$(pwd)
# Paketin kaynak kodlarının olduğu konum
SOURCEDIR="/tmp/kly/build/${name}-${version}"

# initsetup
# derleme dizini yoksa oluşturuluyor
mkdir -p  $ROOTBUILDDIR
# içeriği temizleniyor
rm -rf $ROOTBUILDDIR/*
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 $BUILDDIR

# 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 $SOURCEDIR
make DESTDIR=$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
```

Yukarıdaki kodların sorunsuz çalışabilmesi için ek dosyayalara ihtiyaç vardır. Bu ek dosyaları indirmek için tıklayınız.

tar dosyasını indirdikten sonra istediğiniz bir konumda **bzip2** adında bir dizin oluşturun ve tar dosyasını oluşturulan dizin içinde açınınız.

Paket adında(bzip2) istediğiniz bir konumda bir dizin oluşturun ve dizin içine giriniz. Yukarı verilen script kodlarını build adında bir dosya oluşturup içine kopyalayın ve kaydedin. Daha

sonra build scriptini çalıştırın. Nasıl çalıştırılacağı aşağıdaki komutlarla gösterilmiştir. Aşağıda gösterilen komutları paket için oluşturulan dizinin içinde terminal açarak çalıştırınız.

```
chmod 755 build
fakeroot ./build
```

elfutils

elfutils, ELF dosyalarının oluşturulması, düzenlenmesi ve incelenmesi için gerekli araçları sağlayan bir yazılım paketidir. Bu paket, özellikle derleyiciler ve bağlantı editörleri tarafından üretilen ikili dosyaların yapısını anlamak ve analiz etmek için kullanılır.

Paket, readelf, objdump, eu-strip gibi araçları içerir. Örneğin, readelf komutu, bir ELF dosyasının içeriğini detaylı bir şekilde görüntülemeye olanak tanırken, objdump ise ikili dosyaların iç yapısını analiz etmek için kullanılır. Bu araçlar, geliştiricilerin yazılımlarını optimize etmelerine ve hata ayıklama süreçlerini kolaylaştırmalarına yardımcı olur.

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"
group="dev.libs"

# Paketin yükleneceği tasarlanan sistem konumu
DESTDIR="$HOME/distro/rootfs"
# Derleme konumu
ROOTBUILDDIR="/tmp/kly/build"
# Derleme yapılan paketin derleme konumun
BUILDDIR="/tmp/kly/build/build-${name}-${version}"
# paketin derleme talimatının verildiği konum
PACKAGEDIR=$(pwd)
# Paketin kaynak kodlarının olduğu konum
SOURCEDIR="/tmp/kly/build/${name}-${version}"

# initsetup
# derleme dizini yoksa oluşturuluyor
mkdir -p  $ROOTBUILDDIR
# içeriği temizleniyor
rm -rf $ROOTBUILDDIR/*
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 $BUILDDIR

# 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
```

Paket adında(elfutils) istediğiniz bir konumda bir dizin oluşturun ve dizin içine giriniz. Yukarı verilen script kodlarını build adında bir dosya oluşturup içine kopyalayın ve kaydedin. Daha sonra build scriptini çalıştırın. Nasıl çalıştırılacağı aşağıdaki komutlarla gösterilmiştir. Aşağıda gösterilen komutları paket için oluşturulan dizinin içinde terminal açarak çalıştırınız.

```
chmod 755 build
fakeroot ./build
```

libselinux

libselinux paketi, Linux işletim sistemlerinde güvenlik politikalarının uygulanmasına yardımcı olan bir kütüphanedir. Bu kütüphane, SELinux (Security-Enhanced Linux) mekanizmasının temel bileşenlerinden biridir.

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"
groups="app.arch"

# Paketin yükleneceği tasarlanan sistem konumu
DESTDIR="$HOME/distro/rootfs"
# Derleme konumu
ROOTBUILDDIR="/tmp/kly/build"
# Derleme yapılan paketin derleme konumun
BUILDDIR="/tmp/kly/build/build-${name}-${version}"
# paketin derleme talimatının verildiği konum
PACKAGEDIR=$(pwd)
# Paketin kaynak kodlarının olduğu konum
SOURCEDIR="/tmp/kly/build/${name}-${version}"

# initsetup
# derleme dizini yoksa oluşturuluyor
mkdir -p  $ROOTBUILDDIR
# içeriği temizleniyor
rm -rf $ROOTBUILDDIR/*
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 $BUILDDIR

# setup
cp -prvf $PACKAGEDIR/files/ $SOURCEDIR

# build
patch -Np1 -i files/lfs64.patch
make FTS_LDLIBS="-lfts"

# package
make install DESTDIR=$DESTDIR
```

Yukarıdaki kodların sorunsuz çalışabilmesi için ek dosyayalara ihtiyaç vardır. Bu ek dosyaları indirmek için tıklayınız.

tar dosyasını indirdikten sonra istediğiniz bir konumda **libselinux** adında bir dizin oluşturun ve tar dosyasını oluşturulan dizin içinde açınınız.

Paket adında(libselinux) istediğiniz bir konumda bir dizin oluşturun ve dizin içine giriniz. Yukarı verilen script kodlarını build adında bir dosya oluşturup içine kopyalayın ve kaydedin. Daha sonra build scriptini çalıştırın. Nasıl çalıştırılacağı aşağıdaki komutlarla gösterilmiştir. Aşağıda gösterilen komutları paket için oluşturulan dizinin içinde terminal açarak çalıştırınız.

```
chmod 755 build
fakeroot ./build
```

tar

Tar paketi, Unix ve Linux sistemlerinde dosya arşivleme ve sıkıştırma işlemleri için kullanılan bir dosya formatıdır. "tar" kelimesi, "tape archive" ifadesinin kısaltmasıdır ve başlangıçta manyetik bantlarda veri saklamak amacıyla geliştirilmiştir.

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=""
groups="app.arch"

# Paketin yükleneceği tasarlanan sistem konumu
DESTDIR="$HOME/distro/rootfs"
# Derleme konumu
ROOTBUILDDIR="/tmp/kly/build"
# Derleme yapılan paketin derleme konumun
BUILDDIR="/tmp/kly/build/build-${name}-${version}"
# paketin derleme talimatının verildiği konum
PACKAGEDIR=$(pwd)
# Paketin kaynak kodlarının olduğu konum
SOURCEDIR="/tmp/kly/build/${name}-${version}"

# initsetup
# derleme dizini yoksa oluşturuluyor
mkdir -p  $ROOTBUILDDIR
# içeriği temizleniyor
rm -rf $ROOTBUILDDIR/*
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 $BUILDDIR

# 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
```

Paket adında(tar) istediğiniz bir konumda bir dizin oluşturun ve dizin içine giriniz. Yukarı verilen script kodlarını build adında bir dosya oluşturup içine kopyalayın ve kaydedin. Daha sonra build scriptini çalıştırın. Nasıl çalıştırılacağı aşağıdaki komutlarla gösterilmiştir. Aşağıda gösterilen komutları paket için oluşturulan dizinin içinde terminal açarak çalıştırınız.

```
chmod 755 build
fakeroot ./build
```

zlib

Zlib, veri sıkıştırma ve açma işlemleri için geliştirilmiş açık kaynaklı bir kütüphanedir. Genellikle, Gzip ve PNG gibi formatların temelini oluşturur. Zlib, Deflate algoritmasını kullanarak verileri sıkıştırır ve bu sayede dosya boyutunu önemli ölçüde azaltır. Bu özellik, özellikle ağ üzerinden veri transferi sırasında bant genişliğinden tasarruf sağlamak için oldukça faydalıdır.

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"
groups="app.arch"

# Paketin yükleneceği tasarlanan sistem konumu
DESTDIR="$HOME/distro/rootfs"
# Derleme konumu
ROOTBUILDDIR="/tmp/kly/build"
# Derleme yapılan paketin derleme konumun
BUILDDIR="/tmp/kly/build/build-${name}-${version}"
# paketin derleme talimatının verildiği konum
PACKAGEDIR=$(pwd)
# Paketin kaynak kodlarının olduğu konum
SOURCEDIR="/tmp/kly/build/${name}-${version}"

# initsetup
# derleme dizini yoksa oluşturuluyor
mkdir -p  $ROOTBUILDDIR
# içeriği temizleniyor
rm -rf $ROOTBUILDDIR/*
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 $BUILDDIR

# setup
./configure --prefix=/usr

# build
make

# package
make install pkgconfigdir="/usr/lib64/pkgconfig" DESTDIR=$DESTDIR
```

Paket adında(zlib) istediğiniz bir konumda bir dizin oluşturun ve dizin içine giriniz. Yukarı verilen script kodlarını build adında bir dosya oluşturup içine kopyalayın ve kaydedin. Daha sonra build scriptini çalıştırın. Nasıl çalıştırılacağı aşağıdaki komutlarla gösterilmiştir. Aşağıda gösterilen komutları paket için oluşturulan dizinin içinde terminal açarak çalıştırınız.

```
chmod 755 build
fakeroot ./build
```

brotli

Brotli, Google tarafından geliştirilen ve özellikle web tarayıcıları için optimize edilmiş bir veri sıkıştırma algoritmasıdır. Bu algoritma, HTTP/2 ve HTTPS protokolleri ile birlikte kullanıldığında, web sayfalarının daha hızlı yüklenmesine olanak tanır. Brotli, gzip'e göre daha yüksek sıkıştırma oranları sunarak, veri transferini optimize eder.

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"
groups="sys.apps"

# Paketin yükleneceği tasarlanan sistem konumu
DESTDIR="$HOME/distro/rootfs"
# Derleme konumu
ROOTBUILDDIR="/tmp/kly/build"
# Derleme yapılan paketin derleme konumun
BUILDDIR="/tmp/kly/build/build-${name}-${version}"
# paketin derleme talimatının verildiği konum
PACKAGEDIR=$(pwd)
# Paketin kaynak kodlarının olduğu konum
SOURCEDIR="/tmp/kly/build/${name}-${version}"

# initsetup
# derleme dizini yoksa oluşturuluyor
mkdir -p  $ROOTBUILDDIR
# içeriği temizleniyor
rm -rf $ROOTBUILDDIR/*
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 $BUILDDIR

# 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/
```

Paket adında(brotli) istediğiniz bir konumda bir dizin oluşturun ve dizin içine giriniz. Yukarı verilen script kodlarını build adında bir dosya oluşturup içine kopyalayın ve kaydedin. Daha sonra build scriptini çalıştırın. Nasıl çalıştırılacağı aşağıdaki komutlarla gösterilmiştir. Aşağıda gösterilen komutları paket için oluşturulan dizinin içinde terminal açarak çalıştırınız.

```
chmod 755 build
fakeroot ./build
```

curl

Curl, "Client URL" ifadesinin kısaltmasıdır ve internet protokolleri üzerinden veri transferi yapabilen bir komut satırı aracıdır. Özellikle HTTP, HTTPS, FTP gibi protokollerle çalışarak, web sunucularına istek göndermek ve yanıt almak için kullanılır. Linux sistemlerinde yaygın olarak kullanılan bu araç, geliştiricilere ve sistem yöneticilerine API'lerle etkileşim kurma, dosya indirme veya yükleme gibi işlemleri kolaylaştırır.

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"
groups="net.misc"

# Paketin yükleneceği tasarlanan sistem konumu
DESTDIR="$HOME/distro/rootfs"
# Derleme konumu
ROOTBUILDDIR="/tmp/kly/build"
# Derleme yapılan paketin derleme konumun
BUILDDIR="/tmp/kly/build/build-${name}-${version}"
# paketin derleme talimatının verildiği konum
PACKAGEDIR=$(pwd)
# Paketin kaynak kodlarının olduğu konum
SOURCEDIR="/tmp/kly/build/${name}-${version}"

# initsetup
# derleme dizini yoksa oluşturuluyor
mkdir -p  $ROOTBUILDDIR
# içeriği temizleniyor
rm -rf $ROOTBUILDDIR/*
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 $BUILDDIR

# 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)
./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
```

Paket adında(curl) istediğiniz bir konumda bir dizin oluşturun ve dizin içine giriniz. Yukarı verilen script kodlarını build adında bir dosya oluşturup içine kopyalayın ve kaydedin. Daha sonra build scriptini çalıştırın. Nasıl çalıştırılacağı aşağıdaki komutlarla gösterilmiştir. Aşağıda gösterilen komutları paket için oluşturulan dizinin içinde terminal açarak çalıştırınız.

```
chmod 755 build
fakeroot ./build
```

shadow

Shadow paketi, Linux işletim sistemlerinde kullanıcı hesaplarının şifrelerini güvenli bir şekilde saklamak için kullanılan bir mekanizmadır. Bu paket, kullanıcı bilgilerini ve şifrelerini içeren dosyaların yönetiminde önemli bir rol oynamaktadır.

Derleme

Debian ortamında bu paketin derlenmesi için; **sudo apt install libreadline-dev libcap-dev libcap2-bin** 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"
group="sys.apps"

# Paketin yükleneceği tasarlanan sistem konumu
DESTDIR="$HOME/distro/rootfs"
# Derleme konumu
ROOTBUILDDIR="/tmp/kly/build"
# Derleme yapılan paketin derleme konumun
BUILDDIR="/tmp/kly/build/build-${name}-${version}"
# paketin derleme talimatının verildiği konum
PACKAGEDIR=$(pwd)
# Paketin kaynak kodlarının olduğu konum
SOURCEDIR="/tmp/kly/build/${name}-${version}"

# initsetup
# derleme dizini yoksa oluşturuluyor
mkdir -p  $ROOTBUILDDIR
# içeriği temizleniyor
rm -rf $ROOTBUILDDIR/*
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 $BUILDDIR

# setup
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
```

Yukarıdaki kodların sorunsuz çalışabilmesi için ek dosyayalara ihtiyaç vardır. Bu ek dosyaları indirmek için tıklayınız.

tar dosyasını indirdikten sonra istediğiniz bir konumda **shadow** adında bir dizin oluşturun ve tar dosyasını oluşturulan dizin içinde açınınız.

Paket adında(shadow) istediğiniz bir konumda bir dizin oluşturun ve dizin içine giriniz. Yukarı verilen script kodlarını build adında bir dosya oluşturup içine kopyalayın ve kaydedin. Daha sonra build scriptini çalıştırın. Nasıl çalıştırılacağı aşağıdaki komutlarla gösterilmiştir. Aşağıda gösterilen komutları paket için oluşturulan dizinin içinde terminal açarak çalıştırınız.

```
chmod 755 build
fakeroot ./build
```

file

Linux'ta "file" komutu, dosyaların türlerini tanımlamak için kullanılan bir araçtır. Bu komut, dosyanın içeriğini analiz ederek, dosyanın ne tür bir veri içerdiğini belirler. Örneğin, bir dosyanın metin dosyası mı, ikili dosya mı yoksa bir resim dosyası mı olduğunu tespit edebilir.

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")
group=(sys.apps)
depends=""

# Paketin yükleneceği tasarlanan sistem konumu
DESTDIR="$HOME/distro/rootfs"
# Derleme konumu
ROOTBUILDDIR="/tmp/kly/build"
# Derleme yapılan paketin derleme konumun
BUILDDIR="/tmp/kly/build/build-${name}-${version}"
# paketin derleme talimatının verildiği konum
PACKAGEDIR=$(pwd)
# Paketin kaynak kodlarının olduğu konum
SOURCEDIR="/tmp/kly/build/${name}-${version}"

# initsetup
# derleme dizini yoksa oluşturuluyor
mkdir -p  $ROOTBUILDDIR
# içeriği temizleniyor
rm -rf $ROOTBUILDDIR/*
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 $BUILDDIR

# setup
./configure --prefix=/usr --libdir=/usr/lib64 --enable-static --enable-elf --enable-elf-core \
--enable-zlib --enable-xzlib --enable-bzlib --enable-libseccomp

# build
make

# package
make install DESTDIR=$DESTDIR
```

Paket adında(file) istediğiniz bir konumda bir dizin oluşturun ve dizin içine giriniz. Yukarı verilen script kodlarını build adında bir dosya oluşturup içine kopyalayın ve kaydedin. Daha sonra build scriptini çalıştırın. Nasıl çalıştırılacağı aşağıdaki komutlarla gösterilmiştir. Aşağıda gösterilen komutları paket için oluşturulan dizinin içinde terminal açarak çalıştırınız.

```
chmod 755 build
fakeroot ./build
```

eudev

Eudev, Linux tabanlı sistemlerde cihaz yönetimi için kullanılan bir kullanıcı alanı aracı olan udev'in bir fork'udur. Udev, sistemdeki donanım bileşenlerinin tanınması, yönetilmesi ve olay bildirimlerinin gerçekleştirilmesi için kritik bir rol oynar.

Derleme

Debian ortamında bu paketin derlenmesi için; **sudo apt install libkmod-dev l libgperf-dev** komutuyla paketin 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"
groups="sys.fs"

# Paketin yükleneceği tasarlanan sistem konumu
DESTDIR="$HOME/distro/rootfs"
# Derleme konumu
ROOTBUILDDIR="/tmp/kly/build"
# Derleme yapılan paketin derleme konumun
BUILDDIR="/tmp/kly/build/build-${name}-${version}"
# paketin derleme talimatının verildiği konum
PACKAGEDIR=$(pwd)
# Paketin kaynak kodlarının olduğu konum
SOURCEDIR="/tmp/kly/build/${name}-${version}"

# initsetup
# derleme dizini yoksa oluşturuluyor
mkdir -p  $ROOTBUILDDIR
# içeriği temizleniyor
rm -rf $ROOTBUILDDIR/*
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 $BUILDDIR

# 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
```

Yukarıdaki kodların sorunsuz çalışabilmesi için ek dosyayalara ihtiyaç vardır. Bu ek dosyaları indirmek için tıklayınız.

tar dosyasını indirdikten sonra istediğiniz bir konumda **eudev** adında bir dizin oluşturun ve tar dosyasını oluşturulan dizin içinde açınınız.

Paket adında(eudev) istediğiniz bir konumda bir dizin oluşturun ve dizin içine giriniz. Yukarı verilen script kodlarını build adında bir dosya oluşturup içine kopyalayın ve kaydedin. Daha sonra build scriptini çalıştırın. Nasıl çalıştırılacağı aşağıdaki komutlarla gösterilmiştir. Aşağıda gösterilen komutları paket için oluşturulan dizinin içinde terminal açarak çalıştırınız.

```
chmod 755 build
fakeroot ./build
```

cpio

CPIO (Copy In and Out), Unix ve Linux işletim sistemlerinde dosyaları arşivlemek ve taşımak için kullanılan bir komut satırı aracıdır. CPIO, dosyaları bir arşiv dosyası içinde saklayarak, bu dosyaların daha sonra kolayca geri yüklenmesini sağlar. CPIO, genellikle tarayıcılar ve diğer arşivleme araçları ile birlikte 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"
group="app.arch"

# Paketin yükleneceği tasarlanan sistem konumu
DESTDIR="$HOME/distro/rootfs"
# Derleme konumu
ROOTBUILDDIR="/tmp/kly/build"
# Derleme yapılan paketin derleme konumun
BUILDDIR="/tmp/kly/build/build-${name}-${version}"
# paketin derleme talimatının verildiği konum
PACKAGEDIR=$(pwd)
# Paketin kaynak kodlarının olduğu konum
SOURCEDIR="/tmp/kly/build/${name}-${version}"

# initsetup
# derleme dizini yoksa oluşturuluyor
mkdir -p  $ROOTBUILDDIR
# içeriği temizleniyor
rm -rf $ROOTBUILDDIR/*
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 $BUILDDIR

# setup
CFLAGS+=' -fcommon'
./configure --prefix=/usr

# build
make

# package
make install DESTDIR=$DESTDIR
```

Paket adında(cpio) istediğiniz bir konumda bir dizin oluşturun ve dizin içine giriniz. Yukarı verilen script kodlarını build adında bir dosya oluşturup içine kopyalayın ve kaydedin. Daha sonra build scriptini çalıştırın. Nasıl çalıştırılacağı aşağıdaki komutlarla gösterilmiştir. Aşağıda gösterilen komutları paket için oluşturulan dizinin içinde terminal açarak çalıştırınız.

```
chmod 755 build
fakeroot ./build
```

libsepol

libsepol, SELinux'un güvenlik politikalarını oluşturmak, düzenlemek ve uygulamak için gerekli olan temel bir kütüphanedir. SELinux, Linux çekirdeği üzerinde güvenlik katmanları ekleyerek sistemin güvenliğini artırmayı amaçlar. libsepol, bu güvenlik politikalarının tanımlanması ve yönetilmesi için gerekli olan araçları sağlar.

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"
groups="app.arch"

# Paketin yükleneceği tasarlanan sistem konumu
DESTDIR="$HOME/distro/rootfs"
# Derleme konumu
ROOTBUILDDIR="/tmp/kly/build"
# Derleme yapılan paketin derleme konumun
BUILDDIR="/tmp/kly/build/build-${name}-${version}"
# paketin derleme talimatının verildiği konum
PACKAGEDIR=$(pwd)
# Paketin kaynak kodlarının olduğu konum
SOURCEDIR="/tmp/kly/build/${name}-${version}"

# initsetup
# derleme dizini yoksa oluşturuluyor
mkdir -p  $ROOTBUILDDIR
# içeriği temizleniyor
rm -rf $ROOTBUILDDIR/*
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 $BUILDDIR

# setup
# build
make

# package
make install DESTDIR=$DESTDIR
```

Paket adında(libsepol) istediğiniz bir konumda bir dizin oluşturun ve dizin içine giriniz. Yukarı verilen script kodlarını build adında bir dosya oluşturup içine kopyalayın ve kaydedin. Daha sonra build scriptini çalıştırın. Nasıl çalıştırılacağı aşağıdaki komutlarla gösterilmiştir. Aşağıda gösterilen komutları paket için oluşturulan dizinin içinde terminal açarak çalıştırınız.

```
chmod 755 build
fakeroot ./build
```

kmod

Bash, Linux ve diğer Unix tabanlı işletim sistemlerinde kullanılan bir kabuk programlama dilidir. Kullanıcıların komutlar vererek işletim sistemini yönetmelerine olanak tanır. Bash, kullanıcıların işlemleri otomatikleştirmesine ve betik dosyaları oluşturmasına olanak tanır. Özellikle sistem yöneticileri ve geliştiriciler arasında yaygın olarak kullanılan güçlü bir araçtır.

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-$version.tar.xz"
depends="zlib,xz-utils"
group=(sys.apps)
export PATH=$HOME:$PATH

# Paketin yükleneceği tasarlanan sistem konumu
DESTDIR="$HOME/distro/rootfs"
# Derleme konumu
ROOTBUILDDIR="/tmp/kly/build"
# Derleme yapılan paketin derleme konumun
BUILDDIR="/tmp/kly/build/build-${name}-${version}"
# paketin derleme talimatının verildiği konum
PACKAGEDIR=$(pwd)
# Paketin kaynak kodlarının olduğu konum
SOURCEDIR="/tmp/kly/build/${name}-${version}"

# initsetup
# derleme dizini yoksa oluşturuluyor
mkdir -p  $ROOTBUILDDIR
# içeriği temizleniyor
rm -rf $ROOTBUILDDIR/*
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 $BUILDDIR

# 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
```

Paket adında(kmod) istediğiniz bir konumda bir dizin oluşturun ve dizin içine giriniz. Yukarı verilen script kodlarını build adında bir dosya oluşturup içine kopyalayın ve kaydedin. Daha sonra build scriptini çalıştırın. Nasıl çalıştırılacağı aşağıdaki komutlarla gösterilmiştir. Aşağıda gösterilen komutları paket için oluşturulan dizinin içinde terminal açarak çalıştırınız.

```
chmod 755 build
fakeroot ./build
```

Linux çekirdeği ile donanım arasındaki haberleşmeyi sağlayan kod parçalarıdır. Bu kod parçalarını kernele eklediğimizde kerneli tekrardan derlememiz gerekmektedir. Her kod ekleme ve her kod çıkartma işleminden sonra kernel derlemek ciddi bir iş yükü ve karmaşa oluşturacaktır.

Bu sorunların çözümü için modul vardır. Moduller kernele istediğimiz kod parçalarını ekleme ya da çıkartma yapabilmemizi sağlar. Bu işlemleri yaparken kernel derleme işlemi yapmamıza gerek yoktur.

kmod Komutları

- **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.

Test Edilmesi

Bir modül eklendiğinde veya çıkartıldığında modülle ilgili mesajları dmesg logları ile görebiliriz.

audit

Audit paketi, Linux sistemlerinde güvenlik denetimlerini gerçekleştirmek için tasarlanmış bir yazılımdır. Bu paket, sistemdeki önemli olayları, kullanıcı aktivitelerini ve dosya erişimlerini kaydederek, sistem yöneticilerine kapsamlı bir denetim ve izleme imkanı sunar.

Derleme

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"
groups="sys.process"

# Paketin yükleneceği tasarlanan sistem konumu
DESTDIR="$HOME/distro/rootfs"
# Derleme konumu
ROOTBUILDDIR="/tmp/kly/build"
# Derleme yapılan paketin derleme konumun
BUILDDIR="/tmp/kly/build/build-${name}-${version}"
# paketin derleme talimatının verildiği konum
PACKAGEDIR=$(pwd)
# Paketin kaynak kodlarının olduğu konum
SOURCEDIR="/tmp/kly/build/${name}-${version}"

# initsetup
# derleme dizini yoksa oluşturuluyor
mkdir -p  $ROOTBUILDDIR
# içeriği temizleniyor
rm -rf $ROOTBUILDDIR/*
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 $BUILDDIR

# 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
```

Yukarıdaki kodların sorunsuz çalışabilmesi için ek dosyayalara ihtiyaç vardır. Bu ek dosyaları indirmek için tıklayınız.

tar dosyasını indirdikten sonra istediğiniz bir konumda **audit** adında bir dizin oluşturun ve tar dosyasını oluşturulan dizin içinde açınınız.

Paket adında(audit) istediğiniz bir konumda bir dizin oluşturun ve dizin içine giriniz. Yukarı verilen script kodlarını build adında bir dosya oluşturup içine kopyalayın ve kaydedin. Daha

sonra build scriptini çalıştırın. Nasıl çalıştırılacağı aşağıdaki komutlarla gösterilmiştir. Aşağıda gösterilen komutları paket için oluşturulan dizinin içinde terminal açarak çalıştırınız.

```
chmod 755 build
fakeroot ./build
```

libxcrypt

libxcrypt, Unix benzeri sistemlerde parolaların şifrelenmesi için kullanılan bir kütüphanedir. Bu kütüphane, geleneksel crypt() fonksiyonunun yerini alarak daha güvenli ve esnek bir şifreleme yöntemi sunar. libxcrypt, çeşitli şifreleme algoritmalarını destekler; bunlar arasında bcrypt, scrypt ve Argon2 gibi modern algoritmalar bulunmaktadır.

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")
group=(net.libs)
depends=""

# Paketin yükleneceği tasarlanan sistem konumu
DESTDIR="$HOME/distro/rootfs"
# Derleme konumu
ROOTBUILDDIR="/tmp/kly/build"
# Derleme yapılan paketin derleme konumun
BUILDDIR="/tmp/kly/build/build-${name}-${version}"
# paketin derleme talimatının verildiği konum
PACKAGEDIR=$(pwd)
# Paketin kaynak kodlarının olduğu konum
SOURCEDIR="/tmp/kly/build/${name}-${version}"

# initsetup
# derleme dizini yoksa oluşturuluyor
mkdir -p  $ROOTBUILDDIR
# içeriği temizleniyor
rm -rf $ROOTBUILDDIR/*
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 $BUILDDIR

# setup
./configure --prefix=/usr --libdir=/usr/lib64/ --sbindir=/usr/bin

# build
make -C $SOURCEDIR

# package
make -C $SOURCEDIR install DESTDIR=$DESTDIR
```

Paket adında(libxcrypt) istediğiniz bir konumda bir dizin oluşturun ve dizin içine giriniz. Yukarı verilen script kodlarını build adında bir dosya oluşturup içine kopyalayın ve kaydedin. Daha sonra build scriptini çalıştırın. Nasıl çalıştırılacağı aşağıdaki komutlarla gösterilmiştir. Aşağıda gösterilen komutları paket için oluşturulan dizinin içinde terminal açarak çalıştırınız.

```
chmod 755 build
fakeroot ./build
```

libnsl

libnsl, "Network Services Library" anlamına gelen bir kütüphanedir ve genellikle Unix sistemlerinde ağ hizmetleri ile ilgili işlevsellik sağlamak amacıyla kullanılır. Bu kütüphane, uzaktan prosedür çağrıları (RPC) ve diğer ağ iletişim protokollerinin uygulanmasında kritik bir bileşendir. libnsl, özellikle eski sistemlerde ve uygulamalarda yaygın olarak bulunur ve modern sistemlerde de bazı uygulamalar için gereklidir.

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"
group="net.libs"

# Paketin yükleneceği tasarlanan sistem konumu
DESTDIR="$HOME/distro/rootfs"
# Derleme konumu
ROOTBUILDDIR="/tmp/kly/build"
# Derleme yapılan paketin derleme konumun
BUILDDIR="/tmp/kly/build/build-${name}-${version}"
# paketin derleme talimatının verildiği konum
PACKAGEDIR=$(pwd)
# Paketin kaynak kodlarının olduğu konum
SOURCEDIR="/tmp/kly/build/${name}-${version}"

# initsetup
# derleme dizini yoksa oluşturuluyor
mkdir -p  $ROOTBUILDDIR
# içeriği temizleniyor
rm -rf $ROOTBUILDDIR/*
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 $BUILDDIR

# setup
./configure --prefix=/usr --libdir=/usr/lib64

# build
make $jobs

# package
make install DESTDIR=$DESTDIR
```

Paket adında(libnsl) istediğiniz bir konumda bir dizin oluşturun ve dizin içine giriniz. Yukarı verilen script kodlarını build adında bir dosya oluşturup içine kopyalayın ve kaydedin. Daha sonra build scriptini çalıştırın. Nasıl çalıştırılacağı aşağıdaki komutlarla gösterilmiştir. Aşağıda gösterilen komutları paket için oluşturulan dizinin içinde terminal açarak çalıştırınız.

```
chmod 755 build
fakeroot ./build
```

libbsd

libbsd, geliştiricilere daha güvenli ve taşınabilir kod yazma imkanı tanırken, aynı zamanda BSD sistemlerinde yaygın olarak kullanılan işlevlerin Linux üzerinde de kullanılmasını sağlar. Bu kütüphane, sistem programlama ve ağ programlama gibi alanlarda sıkça tercih edilmektedir.

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=""
group="dev.libs"

# Paketin yükleneceği tasarlanan sistem konumu
DESTDIR="$HOME/distro/rootfs"
# Derleme konumu
ROOTBUILDDIR="/tmp/kly/build"
# Derleme yapılan paketin derleme konumun
BUILDDIR="/tmp/kly/build/build-${name}-${version}"
# paketin derleme talimatının verildiği konum
PACKAGEDIR=$(pwd)
# Paketin kaynak kodlarının olduğu konum
SOURCEDIR="/tmp/kly/build/${name}-${version}"

# initsetup
# derleme dizini yoksa oluşturuluyor
mkdir -p  $ROOTBUILDDIR
# içeriği temizleniyor
rm -rf $ROOTBUILDDIR/*
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 $BUILDDIR

# setup
autoreconf -fvi
./configure --prefix=/usr --libdir=/usr/lib64

# build
make

# package
make install DESTDIR=$DESTDIR
```

Paket adında(libbsd) istediğiniz bir konumda bir dizin oluşturun ve dizin içine giriniz. Yukarı verilen script kodlarını build adında bir dosya oluşturup içine kopyalayın ve kaydedin. Daha sonra build scriptini çalıştırın. Nasıl çalıştırılacağı aşağıdaki komutlarla gösterilmiştir. Aşağıda gösterilen komutları paket için oluşturulan dizinin içinde terminal açarak çalıştırınız.

```
chmod 755 build
fakeroot ./build
```

libtirpc

libtirpc, UNIX sistemlerinde yaygın olarak kullanılan bir RPC kütüphanesidir. Bu kütüphane, istemci ve sunucu uygulamaları arasında uzaktan prosedür çağrıları yapmayı mümkün kılar. libtirpc, özellikle dağıtık sistemlerde veri paylaşımını ve iletişimi kolaylaştırmak amacıyla geliştirilmiştir.

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=""
group="net.libs"

# Paketin yükleneceği tasarlanan sistem konumu
DESTDIR="$HOME/distro/rootfs"
# Derleme konumu
ROOTBUILDDIR="/tmp/kly/build"
# Derleme yapılan paketin derleme konumun
BUILDDIR="/tmp/kly/build/build-${name}-${version}"
# paketin derleme talimatının verildiği konum
PACKAGEDIR=$(pwd)
# Paketin kaynak kodlarının olduğu konum
SOURCEDIR="/tmp/kly/build/${name}-${version}"

# initsetup
# derleme dizini yoksa oluşturuluyor
mkdir -p  $ROOTBUILDDIR
# içeriği temizleniyor
rm -rf $ROOTBUILDDIR/*
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 $BUILDDIR

# setup
./configure --prefix=/usr --libdir=/usr/lib64 --sysconfdir=/etc --disable-gssapi

# build
make

# package
make install DESTDIR=$DESTDIR
```

Paket adında(libtirpc) istediğiniz bir konumda bir dizin oluşturun ve dizin içine giriniz. Yukarı verilen script kodlarını build adında bir dosya oluşturup içine kopyalayın ve kaydedin. Daha sonra build scriptini çalıştırın. Nasıl çalıştırılacağı aşağıdaki komutlarla gösterilmiştir. Aşağıda gösterilen komutları paket için oluşturulan dizinin içinde terminal açarak çalıştırınız.

```
chmod 755 build
fakeroot ./build
```

e2fsprogs

e2fsprogs, Linux tabanlı sistemlerde yaygın olarak kullanılan bir dosya sistemi yönetim aracıdır. Bu paket, ext2, ext3 ve ext4 dosya sistemleri üzerinde çeşitli işlemler yapabilmek için gerekli olan araçları içerir. Örneğin, dosya sistemi oluşturma, onarma, kontrol etme ve boyutlandırma gibi işlemler e2fsprogs ile gerçekleştirilebilir.

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"
groups="sys.fs"

# Paketin yükleneceği tasarlanan sistem konumu
DESTDIR="$HOME/distro/rootfs"
# Derleme konumu
ROOTBUILDDIR="/tmp/kly/build"
# Derleme yapılan paketin derleme konumun
BUILDDIR="/tmp/kly/build/build-${name}-${version}"
# paketin derleme talimatının verildiği konum
PACKAGEDIR=$(pwd)
# Paketin kaynak kodlarının olduğu konum
SOURCEDIR="/tmp/kly/build/${name}-${version}"

# initsetup
# derleme dizini yoksa oluşturuluyor
mkdir -p  $ROOTBUILDDIR
# içeriği temizleniyor
rm -rf $ROOTBUILDDIR/*
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 $BUILDDIR

# setup
./configure --sbindir=/usr/bin --libdir=/usr/lib64/

# build
make

# package
make install DESTDIR=$DESTDIR
rm -rf $DESTDIR/usr/share/man/man8/fsck.8
```

Paket adında(e2fsprogs) istediğiniz bir konumda bir dizin oluşturun ve dizin içine giriniz. Yukarı verilen script kodlarını build adında bir dosya oluşturup içine kopyalayın ve kaydedin. Daha sonra build scriptini çalıştırın. Nasıl çalıştırılacağı aşağıdaki komutlarla gösterilmiştir. Aşağıda gösterilen komutları paket için oluşturulan dizinin içinde terminal açarak çalıştırınız.

```
chmod 755 build
fakeroot ./build
```

dostools

Dostools, Linux işletim sistemlerinde kullanılan bir dizi araç ve komut setidir. Bu araçlar, sistem yöneticilerine ve geliştiricilere, sistem yönetimi, dosya işlemleri ve ağ yönetimi gibi çeşitli görevleri daha verimli bir şekilde gerçekleştirme imkanı sunar.

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"
groups="sys.block"

# Paketin yükleneceği tasarlanan sistem konumu
DESTDIR="$HOME/distro/rootfs"
# Derleme konumu
ROOTBUILDDIR="/tmp/kly/build"
# Derleme yapılan paketin derleme konumun
BUILDDIR="/tmp/kly/build/build-${name}-${version}"
# paketin derleme talimatının verildiği konum
PACKAGEDIR=$(pwd)
# Paketin kaynak kodlarının olduğu konum
SOURCEDIR="/tmp/kly/build/${name}-${version}"

# initsetup
# derleme dizini yoksa oluşturuluyor
mkdir -p  $ROOTBUILDDIR
# içeriği temizleniyor
rm -rf $ROOTBUILDDIR/*
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 $BUILDDIR

# setup
./autogen.sh
./configure --prefix=/usr --libdir=/usr/lib64/ --enable-compat-symlinks

# build
make

# package
make install DESTDIR=$DESTDIR
```

Paket adında(dostools) istediğiniz bir konumda bir dizin oluşturun ve dizin içine giriniz. Yukarı verilen script kodlarını build adında bir dosya oluşturup içine kopyalayın ve kaydedin. Daha sonra build scriptini çalıştırın. Nasıl çalıştırılacağı aşağıdaki komutlarla gösterilmiştir. Aşağıda gösterilen komutları paket için oluşturulan dizinin içinde terminal açarak çalıştırınız.

```
chmod 755 build
fakeroot ./build
```

initramfs-tools

initramfs-tools, Debian tabanlı sistemlerde kullanılan bir araçtır ve initramfs (initial RAM file system) oluşturmak için kullanılır. Bu araç, sistem açılırken kullanılan geçici bir dosya sistemini oluşturur ve gerekli modülleri yükler. initramfs için farklı araçlarda kullanılabilir.

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"
groups="sys.fs"


# Paketin yükleneceği tasarlanan sistem konumu
DESTDIR="$HOME/distro/rootfs"
# Derleme konumu
ROOTBUILDDIR="/tmp/kly/build"
# Derleme yapılan paketin derleme konumun
BUILDDIR="/tmp/kly/build/build-${name}-${version}"
# paketin derleme talimatının verildiği konum
PACKAGEDIR=$(pwd)
# Paketin kaynak kodlarının olduğu konum
SOURCEDIR="/tmp/kly/build/${name}-${version}"

# initsetup
# derleme dizini yoksa oluşturuluyor
mkdir -p  $ROOTBUILDDIR
# içeriği temizleniyor
rm -rf $ROOTBUILDDIR/*
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 $BUILDDIR

# 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

# 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 sysconf
mkdir -p ${DESTDIR}/etc/sysconf.d
install $SOURCEDIR/initramfs-tools.sysconf ${DESTDIR}/etc/sysconf.d/initramfs-tools
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
```

Yukarıdaki kodların sorunsuz çalışabilmesi için ek dosyayalara ihtiyaç vardır. Bu ek dosyaları indirmek için tıklayınız.

tar dosyasını indirdikten sonra istediğiniz bir konumda **initramfs-tools** adında bir dizin oluşturun ve tar dosyasını oluşturulan dizin içinde açınınız.

Paket adında(initramfs-tools) istediğiniz bir konumda bir dizin oluşturun ve dizin içine giriniz. Yukarı verilen script kodlarını build adında bir dosya oluşturup içine kopyalayın ve kaydedin. Daha sonra build scriptini çalıştırın. Nasıl çalıştırılacağı aşağıdaki komutlarla gösterilmiştir. Aşağıda gösterilen komutları paket için oluşturulan dizinin içinde terminal açarak çalıştırınız.

```
chmod 755 build
fakeroot ./build
```

/etc/initramfs-tools/modules

modules dosyası initrd oluşturulma ve güncelleme durumunda isteğe bağlı olarak modullerin eklenmesisini ve **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.

```
### This file is the template for /etc/initramfs-tools/modules.
### It is not a configuration file itself.
###
# List of modules that you want to include in your initramfs.
# They will be loaded at boot time in the order below.
#
# Syntax:  module_name [args ...]
#
# You must run update-initramfs(8) to effect this change.
#
# Examples:
#
# raid1
# sd_mod
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 oluşturlacaksa, yani chroot ile sisteme erişiliyorsa yukarıdaki komut yeterli olmayacaktır. chroot öncesinde sistemin **dev sys proc run** dizinlerinin

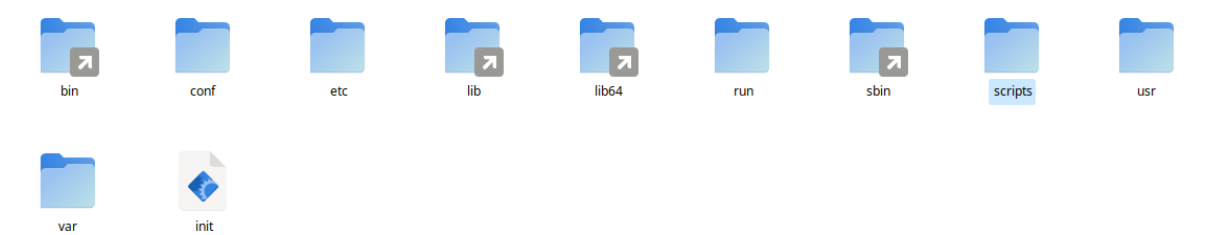
bağlanılması gerekmektedir. Dizindeki sistemimizin dizin konumu **/$HOME/distro/rootfs** olsun. Buna göre aşağıda sisteme yukarıdaki komutu çalıştırmadan önce çalıştırılması gereken komutlar aşağıda verilmiştir. Dikkat edilmesi gereken en önemli noktalardan biriside bu komutlar **root** yetkisiyle çalıştırılmalıdır.

```
rootfs="$HOME/distro/rootfs"
distro="$HOME/distro"
mkdir -p $rootfs/dev
mkdir -p $rootfs/sys
mkdir -p $rootfs/proc
mkdir -p $rootfs/run
mkdir -p $rootfs/tmp
mount --bind /dev $rootfs/dev
mount --bind /sys $rootfs/sys
mount --bind /proc $rootfs/proc
mount --bind /run$rootfs/run
mount --bind /tmp $rootfs/tmp

### 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

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

umount -lf -R $rootfs/dev 2>/dev/null
umount -lf -R $rootfs/sys 2>/dev/null
umount -lf -R $rootfs/proc 2>/dev/null
umount -lf -R $rootfs/run 2>/dev/null
umount -lf -R $rootfs/tmp 2>/dev/null
#### 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

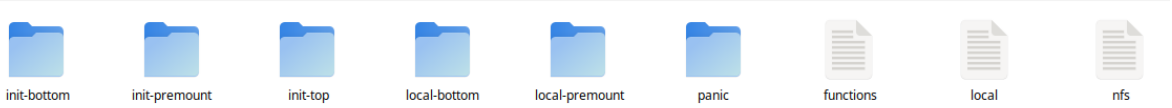
Sistemin açılması için **vmlinuz**, **initrd.img** ve **grub.cfg** dosyalarının olması yeterlidir. **initrd.img** sistemin açılma sürecini yürüten bir kernel yardımcı ön sistemidir. **initrd.img** açıldığında aşğıdaki gibi bir dizin yapısı olur. Bu dizinler içindeki **script** dizini çok önemlidir. Bu 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, özellikle XML ve HTML belgeleri ile çalışmak için tasarlanmış, yüksek performanslı bir C kütüphanesidir. Geliştiricilere, XML belgelerini analiz etme, oluşturma ve düzenleme gibi işlemleri gerçekleştirme imkanı sunar. libxml2, DOM (Document Object Model) ve SAX (Simple API for XML) gibi iki farklı API ile çalışabilme yeteneğine sahiptir. Bu sayede, hem bellek dostu hem de hızlı bir şekilde veri işleme imkanı sağlar.

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"
groups="dev.libs"


# Paketin yükleneceği tasarlanan sistem konumu
DESTDIR="$HOME/distro/rootfs"
# Derleme konumu
ROOTBUILDDIR="/tmp/kly/build"
# Derleme yapılan paketin derleme konumun
BUILDDIR="/tmp/kly/build/build-${name}-${version}"
# paketin derleme talimatının verildiği konum
PACKAGEDIR=$(pwd)
# Paketin kaynak kodlarının olduğu konum
SOURCEDIR="/tmp/kly/build/${name}-${version}"

# initsetup
# derleme dizini yoksa oluşturuluyor
mkdir -p  $ROOTBUILDDIR
# içeriği temizleniyor
rm -rf $ROOTBUILDDIR/*
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 $BUILDDIR

# 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
```

Paket adında(libxml2) istediğiniz bir konumda bir dizin oluşturun ve dizin içine giriniz. Yukarı verilen script kodlarını build adında bir dosya oluşturup içine kopyalayın ve kaydedin. Daha sonra build scriptini çalıştırın. Nasıl çalıştırılacağı aşağıdaki komutlarla gösterilmiştir. Aşağıda gösterilen komutları paket için oluşturulan dizinin içinde terminal açarak çalıştırınız.

```
chmod 755 build
fakeroot ./build
```

expat

Expat, özellikle C dilinde geliştirilmiş bir XML ayrıştırma kütüphanesidir. Bu kütüphane, XML belgelerini okuma ve işleme süreçlerini kolaylaştırmak amacıyla tasarlanmıştır. Expat, olay tabanlı bir ayrıştırma modeli kullanarak, XML belgelerinin içeriğini parçalara ayırır ve bu parçaları işlemek için geliştiricilere bir dizi geri çağırma (callback) fonksiyonu sunar. Bu sayede, büyük XML dosyaları ile çalışırken bellek verimliliği sağlanır.

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=""
    group="dev.libs"

    # Paketin yükleneceği tasarlanan sistem konumu
    DESTDIR="$HOME/distro/rootfs"
    # Derleme konumu
    ROOTBUILDDIR="/tmp/kly/build"
    # Derleme yapılan paketin derleme konumun
    BUILDDIR="/tmp/kly/build/build-${name}-${version}"
    # paketin derleme talimatının verildiği konum
    PACKAGEDIR=$(pwd)
    # Paketin kaynak kodlarının olduğu konum
    SOURCEDIR="/tmp/kly/build/${name}-${version}"

    # initsetup
    # derleme dizini yoksa oluşturuluyor
    mkdir -p  $ROOTBUILDDIR
    # içeriği temizleniyor
    rm -rf $ROOTBUILDDIR/*
    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 $BUILDDIR

    # 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
```

Paket adında(expat) istediğiniz bir konumda bir dizin oluşturun ve dizin içine giriniz. Yukarı verilen script kodlarını build adında bir dosya oluşturup içine kopyalayın ve kaydedin. Daha sonra build scriptini çalıştırın. Nasıl çalıştırılacağı aşağıdaki komutlarla gösterilmiştir. Aşağıda gösterilen komutları paket için oluşturulan dizinin içinde terminal açarak çalıştırınız.

```
chmod 755 build
fakeroot ./build
```

libmd

libmd, özellikle BSD tabanlı sistemlerde bulunan bir kütüphanedir ve çeşitli kriptografik hash fonksiyonlarını (MD5, SHA-1, SHA-256 gibi) destekler. Bu kütüphane, veri bütünlüğünü sağlamak ve şifreleme işlemlerini gerçekleştirmek için kullanılır. Örneğin, bir dosyanın hash değerini hesaplamak, dosyanın değiştirilip değiştirilmediğini kontrol etmek için yaygın bir yöntemdir.

Derleme

```
    #!/usr/bin/env bash
    name="libmd"
    version="1.1.0"
    description="Message Digest functions from BSD systems"
    depends=""
    group="app.crypt"
    source="https://archive.hadrons.org/software/libmd/libmd-$version.tar.xz"

    # Paketin yükleneceği tasarlanan sistem konumu
    DESTDIR="$HOME/distro/rootfs"
    # Derleme konumu
    ROOTBUILDDIR="/tmp/kly/build"
    # Derleme yapılan paketin derleme konumun
    BUILDDIR="/tmp/kly/build/build-${name}-${version}"
    # paketin derleme talimatının verildiği konum
    PACKAGEDIR=$(pwd)
    # Paketin kaynak kodlarının olduğu konum
    SOURCEDIR="/tmp/kly/build/${name}-${version}"

    # initsetup
    # derleme dizini yoksa oluşturuluyor
    mkdir -p  $ROOTBUILDDIR
    # içeriği temizleniyor
    rm -rf $ROOTBUILDDIR/*
    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 $BUILDDIR

    # setup
./configure --prefix=/usr --libdir=/usr/lib64/

    # build
    make

    # package
    make install DESTDIR=$DESTDIR
```

Paket adında(libmd) istediğiniz bir konumda bir dizin oluşturun ve dizin içine giriniz. Yukarı verilen script kodlarını build adında bir dosya oluşturup içine kopyalayın ve kaydedin. Daha sonra build scriptini çalıştırın. Nasıl çalıştırılacağı aşağıdaki komutlarla gösterilmiştir. Aşağıda gösterilen komutları paket için oluşturulan dizinin içinde terminal açarak çalıştırınız.

```
chmod 755 build
fakeroot ./build
```

libaio

Libio, C dilinde girdi/çıktı işlemlerini kolaylaştıran bir kütüphanedir. Temel olarak, dosya okuma ve yazma işlemlerini, bellek yönetimini ve veri akışını yönetmek için kullanılır. Libio, performansı artırmak amacıyla tamponlama (buffering) mekanizmaları kullanır. Bu sayede, verilerin daha verimli bir şekilde işlenmesini sağlar. Örneğin, bir dosyadan veri okurken, veriler önce bir tampon belleğe alınır ve ardından işlenir. Bu, disk erişimlerini azaltarak programın genel performansını artırır.

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=""
group="dev.libs"

# Paketin yükleneceği tasarlanan sistem konumu
DESTDIR="$HOME/distro/rootfs"
# Derleme konumu
ROOTBUILDDIR="/tmp/kly/build"
# Derleme yapılan paketin derleme konumun
BUILDDIR="/tmp/kly/build/build-${name}-${version}"
# paketin derleme talimatının verildiği konum
PACKAGEDIR=$(pwd)
# Paketin kaynak kodlarının olduğu konum
SOURCEDIR="/tmp/kly/build/${name}-${version}"

# initsetup
# derleme dizini yoksa oluşturuluyor
mkdir -p  $ROOTBUILDDIR
# içeriği temizleniyor
rm -rf $ROOTBUILDDIR/*
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 $BUILDDIR

# setup

# build
make

# package
make install DESTDIR=$DESTDIR
```

Paket adında(libio) istediğiniz bir konumda bir dizin oluşturun ve dizin içine giriniz. Yukarı verilen script kodlarını build adında bir dosya oluşturup içine kopyalayın ve kaydedin. Daha sonra build scriptini çalıştırın. Nasıl çalıştırılacağı aşağıdaki komutlarla gösterilmiştir. Aşağıda gösterilen komutları paket için oluşturulan dizinin içinde terminal açarak çalıştırınız.

```
chmod 755 build
fakeroot ./build
```

lvm2

LVM2 (Logical Volume Manager 2), Linux işletim sistemlerinde disk alanını yönetmek için kullanılan bir araçtır. LVM2, fiziksel disklerin mantıksal birimlere dönüştürülmesine olanak tanır. Bu sayede, disk alanı dinamik olarak genişletilebilir veya daraltılabilir. LVM2, sistem yöneticilerine disk alanını daha verimli bir şekilde kullanma imkanı sunar. Örneğin, bir fiziksel disk üzerinde birden fazla mantıksal birim oluşturabilir ve bu birimlerin boyutlarını ihtiyaçlara göre değiştirebilirsiniz.

Debian ortamında derlemek lvm2 paketinde sorunlar çıkmaktadır. Bunun sebebi kernel derlenmesi sırasında lvm ile ilgili özellikleri aktif etmek gerekiyor. Bundan dolayı isteyen aşağıda derleme bölümünde verilen derleme talimatını kullanarak başka dağıtımlarda derleme yapabilir. Burada geliştirilmesi anlatılan sistem üzerinde derlenen paketi indirip hazırladığımız sisteme kuran script aşağıda verilmiştir.

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"

# Paketin yükleneceği tasarlanan sistem konumu
DESTDIR="$HOME/distro/rootfs"
# Derleme konumu
ROOTBUILDDIR="/tmp/kly/build"
# Derleme yapılan paketin derleme konumun
BUILDDIR="/tmp/kly/build/build-${name}-${version}"
# paketin derleme talimatının verildiği konum
PACKAGEDIR=$(pwd)
# Paketin kaynak kodlarının olduğu konum
SOURCEDIR="/tmp/kly/build/${name}-${version}"

# initsetup
# derleme dizini yoksa oluşturuluyor
mkdir -p  $ROOTBUILDDIR
# içeriği temizleniyor
rm -rf $ROOTBUILDDIR/*
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 $BUILDDIR

# setup
mkdir -p $SOURCEDIR
cd $SOURCEDIR
#dosya indiriliyor
wget -O lvm2.kly https://kendilinuxunuyap.github.io/_static/files/lvm2/lvm2-2_03_21.kly
tar -xf lvm2.kly
tar -xf rootfs.tar.xz

# build
#make

# package
cd $SOURCEDIR
cp -prfv etc  ${DESTDIR}/
cp -prfv usr  ${DESTDIR}/
```

Paket adında(lvm2) istediğiniz bir konumda bir dizin oluşturun ve dizin içine giriniz. Yukarı verilen script kodlarını build adında bir dosya oluşturup içine kopyalayın ve kaydedin. Daha sonra build scriptini çalıştırın. Nasıl çalıştırılacağı aşağıdaki komutlarla gösterilmiştir. Aşağıda gösterilen komutları paket için oluşturulan dizinin içinde terminal açarak çalıştırınız.

```
chmod 755 build
fakeroot ./build
```

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=""
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 $SOURCEDIR
}
setup(){
                ./configure --prefix=/usr --libdir=/usr/lib64/  CONFIG_SHELL=/bin/bash --sbindir=/usr/bin \
                --sysconfdir=/etc --localstatedir=/var \
                --enable-cmdlib --enable-dmeventd --enable-lvmpolld --enable-pkgconfig --enable-readline \
                --enable-udev_rules --enable-udev_sync --enable-write_install --disable-systemd \
                --with-cache=internal --with-default-dm-run-dir=/run --with-default-locking-dir=/run/lock/lvm \
                --with-default-pid-dir=/run --with-default-run-dir=/run/lvm --with-thin=internal --with-udev-prefix=/usr
}
build(){
    make
}
package() {
    make install DESTDIR=$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 adında(lvm2) istediğiniz bir konumda bir dizin oluşturun ve dizin içine giriniz. Yukarı verilen script kodlarını build adında bir dosya oluşturup içine kopyalayın ve kaydedin. Daha sonra build scriptini çalıştırın. Nasıl çalıştırılacağı aşağıdaki komutlarla gösterilmiştir. Aşağıda gösterilen komutları paket için oluşturulan dizinin içinde terminal açarak çalıştırınız.

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

popt

Popt, "Portable Options Parsing Library" ifadesinin kısaltmasıdır ve özellikle C programlama dilinde geliştirilmiş bir kütüphanedir. Bu kütüphane, komut satırı argümanlarını analiz etmek ve yönetmek için kullanılır. Popt, kullanıcıların programlarına daha anlaşılır ve esnek bir seçenek yönetimi eklemelerine olanak tanır.

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=""
group="sys.apps"

# Paketin yükleneceği tasarlanan sistem konumu
DESTDIR="$HOME/distro/rootfs"
# Derleme konumu
ROOTBUILDDIR="/tmp/kly/build"
# Derleme yapılan paketin derleme konumun
BUILDDIR="/tmp/kly/build/build-${name}-${version}"
# paketin derleme talimatının verildiği konum
PACKAGEDIR=$(pwd)
# Paketin kaynak kodlarının olduğu konum
SOURCEDIR="/tmp/kly/build/${name}-${version}"

# initsetup
# derleme dizini yoksa oluşturuluyor
mkdir -p  $ROOTBUILDDIR
# içeriği temizleniyor
rm -rf $ROOTBUILDDIR/*
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 $BUILDDIR

# setup
CFLAGS+=" -ffat-lto-objects"
./configure --prefix=/usr

# build
make

# package
make install DESTDIR=$DESTDIR
```

Paket adında(popt) istediğiniz bir konumda bir dizin oluşturun ve dizin içine giriniz. Yukarı verilen script kodlarını build adında bir dosya oluşturup içine kopyalayın ve kaydedin. Daha sonra build scriptini çalıştırın. Nasıl çalıştırılacağı aşağıdaki komutlarla gösterilmiştir. Aşağıda gösterilen komutları paket için oluşturulan dizinin içinde terminal açarak çalıştırınız.

```
chmod 755 build
fakeroot ./build
```

icu

ICU, çok dilli uygulamalar geliştirmek için gerekli olan metin işleme, tarih ve saat formatlama, sayı biçimlendirme gibi işlevleri sağlayan bir kütüphanedir. Unicode standardını destekleyerek, farklı dillerdeki karakterlerin doğru bir şekilde işlenmesini ve görüntülenmesini mümkün kılar. Özellikle, uluslararasılaşma (i18n) ve yerelleştirme (l10n) süreçlerinde kritik bir rol oynar.

Derleme

```
#!/usr/bin/env bash
name="icu"
version="74-2"
description="icu International Components for Unicode"
source=("https://github.com/unicode-org/icu/releases/download/release-${version}/icu4c-74_2-src.tgz")
depends=()
group=(dev.libs)

# Paketin yükleneceği tasarlanan sistem konumu
DESTDIR="$HOME/distro/rootfs"
# Derleme konumu
ROOTBUILDDIR="/tmp/kly/build"
# Derleme yapılan paketin derleme konumun
BUILDDIR="/tmp/kly/build/build-${name}-${version}"
# paketin derleme talimatının verildiği konum
PACKAGEDIR=$(pwd)
# Paketin kaynak kodlarının olduğu konum
SOURCEDIR="/tmp/kly/build/${name}-${version}"

# initsetup
# derleme dizini yoksa oluşturuluyor
mkdir -p  $ROOTBUILDDIR
# içeriği temizleniyor
rm -rf $ROOTBUILDDIR/*
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 $BUILDDIR

# setup
cd source
./configure --prefix=/usr --libdir=/usr/lib64/

# build
make

# package
make install DESTDIR=$DESTDIR
chmod +x "$DESTDIR"/usr/bin/icu-config
```

Paket adında(icu) istediğiniz bir konumda bir dizin oluşturun ve dizin içine giriniz. Yukarı verilen script kodlarını build adında bir dosya oluşturup içine kopyalayın ve kaydedin. Daha sonra build scriptini çalıştırın. Nasıl çalıştırılacağı aşağıdaki komutlarla gösterilmiştir. Aşağıda gösterilen komutları paket için oluşturulan dizinin içinde terminal açarak çalıştırınız.

```
chmod 755 build
fakeroot ./build
```

iproute2

iproute2, Linux tabanlı sistemlerde ağ yönetimi için geliştirilmiş bir araç setidir. Bu araç seti, özellikle ağ yönlendirmesi ve trafik kontrolü gibi karmaşık işlemleri gerçekleştirmek için kullanılır. iproute2, ip komutu ile birlikte gelir ve bu komut, ağ arayüzlerini, yönlendirme tablolarını ve diğer ağ yapılandırmalarını yönetmek için kapsamlı bir arayüz sunar.

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=""
group="sys.apps"

# Paketin yükleneceği tasarlanan sistem konumu
DESTDIR="$HOME/distro/rootfs"
# Derleme konumu
ROOTBUILDDIR="/tmp/kly/build"
# Derleme yapılan paketin derleme konumun
BUILDDIR="/tmp/kly/build/build-${name}-${version}"
# paketin derleme talimatının verildiği konum
PACKAGEDIR=$(pwd)
# Paketin kaynak kodlarının olduğu konum
SOURCEDIR="/tmp/kly/build/${name}-${version}"

# initsetup
# derleme dizini yoksa oluşturuluyor
mkdir -p  $ROOTBUILDDIR
# içeriği temizleniyor
rm -rf $ROOTBUILDDIR/*
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 $BUILDDIR

# 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
```

Yukarıdaki kodların sorunsuz çalışabilmesi için ek dosyayalara ihtiyaç vardır. Bu ek dosyaları indirmek için tıklayınız.

tar dosyasını indirdikten sonra istediğiniz bir konumda **iproute2** adında bir dizin oluşturun ve tar dosyasını oluşturulan dizin içinde açınınız.

Paket adında(iproute2) istediğiniz bir konumda bir dizin oluşturun ve dizin içine giriniz. Yukarı verilen script kodlarını build adında bir dosya oluşturup içine kopyalayın ve kaydedin. Daha sonra build scriptini çalıştırın. Nasıl çalıştırılacağı aşağıdaki komutlarla gösterilmiştir. Aşağıda gösterilen komutları paket için oluşturulan dizinin içinde terminal açarak çalıştırınız.

```
chmod 755 build
fakeroot ./build
```

net-tools

net-tools, Linux tabanlı sistemlerde ağ yönetimi için kullanılan klasik bir araçlar paketidir. Bu paket, ifconfig, route, netstat, arp gibi komutları içerir. ifconfig komutu, ağ arayüzlerinin durumunu görüntülemek ve yapılandırmak için kullanılırken, route komutu yönlendirme tablolarını yönetmekte kullanılır. netstat ise ağ bağlantılarını ve istatistiklerini gösterir.

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=""
group="sys.apps"

# Paketin yükleneceği tasarlanan sistem konumu
DESTDIR="$HOME/distro/rootfs"
# Derleme konumu
ROOTBUILDDIR="/tmp/kly/build"
# Derleme yapılan paketin derleme konumun
BUILDDIR="/tmp/kly/build/build-${name}-${version}"
# paketin derleme talimatının verildiği konum
PACKAGEDIR=$(pwd)
# Paketin kaynak kodlarının olduğu konum
SOURCEDIR="/tmp/kly/build/${name}-${version}"

# initsetup
# derleme dizini yoksa oluşturuluyor
mkdir -p  $ROOTBUILDDIR
# içeriği temizleniyor
rm -rf $ROOTBUILDDIR/*
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 $BUILDDIR

# 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
```

Paket adında(net-tools) istediğiniz bir konumda bir dizin oluşturun ve dizin içine giriniz. Yukarı verilen script kodlarını build adında bir dosya oluşturup içine kopyalayın ve kaydedin. Daha sonra build scriptini çalıştırın. Nasıl çalıştırılacağı aşağıdaki komutlarla gösterilmiştir. Aşağıda gösterilen komutları paket için oluşturulan dizinin içinde terminal açarak çalıştırınız.

```
chmod 755 build
fakeroot ./build
```

dhcp

Bu protokol, bir istemci cihazın ağa bağlandığında, DHCP sunucusuna bir istek göndererek IP adresi talep etmesiyle başlar. Sunucu, istemciye uygun bir IP adresi, alt ağ maskesi, varsayılan ağ geçidi ve DNS sunucusu gibi bilgileri iletir.

Derleme

```
#!/usr/bin/env bash
name="dhcp"
version="4.4.3"
description="GNU regular expression matcher"
source="https://downloads.isc.org/isc/dhcp/4.4.3/dhcp-4.4.3.tar.gz"
depends=""
group="sys.apps"

# Paketin yükleneceği tasarlanan sistem konumu
DESTDIR="$HOME/distro/rootfs"
# Derleme konumu
ROOTBUILDDIR="/tmp/kly/build"
# Derleme yapılan paketin derleme konumun
BUILDDIR="/tmp/kly/build/build-${name}-${version}"
# paketin derleme talimatının verildiği konum
PACKAGEDIR=$(pwd)
# Paketin kaynak kodlarının olduğu konum
SOURCEDIR="/tmp/kly/build/${name}-${version}"

# initsetup
# derleme dizini yoksa oluşturuluyor
mkdir -p  $ROOTBUILDDIR
# içeriği temizleniyor
rm -rf $ROOTBUILDDIR/*
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 $BUILDDIR

# 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
```

Yukarıdaki kodların sorunsuz çalışabilmesi için ek dosyayalara ihtiyaç vardır. Bu ek dosyaları indirmek için tıklayınız.

tar dosyasını indirdikten sonra istediğiniz bir konumda **dhcp** adında bir dizin oluşturun ve tar dosyasını oluşturulan dizin içinde açınınız.

Paket adında(dhcp) istediğiniz bir konumda bir dizin oluşturun ve dizin içine giriniz. Yukarı verilen script kodlarını build adında bir dosya oluşturup içine kopyalayın ve kaydedin. Daha sonra build scriptini çalıştırın. Nasıl çalıştırılacağı aşağıdaki komutlarla gösterilmiştir. Aşağıda gösterilen komutları paket için oluşturulan dizinin içinde terminal açarak çalıştırınız.

```
chmod 755 build
fakeroot ./build
```

openrc

OpenRC, sistem başlangıcını ve hizmetlerin yönetimini sağlamak amacıyla geliştirilmiş bir init sistemidir.

Derleme

```
#!/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=""
group="sys.apps,pam"

# Paketin yükleneceği tasarlanan sistem konumu
DESTDIR="$HOME/distro/rootfs"
# Derleme konumu
ROOTBUILDDIR="/tmp/kly/build"
# Derleme yapılan paketin derleme konumun
BUILDDIR="/tmp/kly/build/build-${name}-${version}"
# paketin derleme talimatının verildiği konum
PACKAGEDIR=$(pwd)
# Paketin kaynak kodlarının olduğu konum
SOURCEDIR="/tmp/kly/build/${name}-${version}"

# initsetup
# derleme dizini yoksa oluşturuluyor
mkdir -p  $ROOTBUILDDIR
# içeriği temizleniyor
rm -rf $ROOTBUILDDIR/*
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 $BUILDDIR

# 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
export DESTDIR=${DESTDIR}//
DESTDIR="$DESTDIR" meson install --no-rebuild -C $BUILDDIR
rm -f ${DESTDIR}/etc/runlevels/*/*          # disable all services
rm ${DESTDIR}//etc/init.d/functions.sh
ln -s ../../lib/rc/sh/functions.sh ${DESTDIR}/etc/init.d/functions.sh
mkdir -p ${DESTDIR}/etc/sysconf.d/          # install sysconf script
install $SOURCEDIR/files/openrc.sysconf ${DESTDIR}/etc/sysconf.d/openrc
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
```

Yukarıdaki kodların sorunsuz çalışabilmesi için ek dosyayalara ihtiyaç vardır. Bu ek dosyaları indirmek için tıklayınız.

Bu extras dosyalarını indirmek için tıklayınız..

tar dosyalarını indirdikten sonra istediğiniz bir konumda **openrc** adında bir dizin oluşturun ve tar dosyasını oluşturulan dizin içinde açınınız.

Paket adında(openrc) istediğiniz bir konumda bir dizin oluşturun ve dizin içine giriniz. Yukarı verilen script kodlarını build adında bir dosya oluşturup içine kopyalayın ve kaydedin. Daha sonra build scriptini çalıştırın. Nasıl çalıştırılacağı aşağıdaki komutlarla gösterilmiştir. Aşağıda gösterilen komutları paket için oluşturulan dizinin içinde terminal açarak çalıştırınız.

```
chmod 755 build
fakeroot ./build
```

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.

Basit kullanım

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.
```

rsync

rsync, dosya transferi ve senkronizasyonu için geliştirilmiş bir yazılımdır. Temel işlevi, yerel veya uzak sistemler arasında dosyaların ve dizinlerin hızlı ve verimli bir şekilde kopyalanmasını sağlamaktır. rsync, yalnızca değişen verileri transfer ederek bant genişliği kullanımını optimize eder. Bu özellik, büyük dosyaların veya dizinlerin güncellenmesi gerektiğinde önemli bir avantaj sunar.

Derleme

Debian ortamında bu paketin derlenmesi için;

- **sudo apt install libzstd-dev libacl1-dev libacl1**

komutuyla paketin 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"
groups="net.misc"

# Paketin yükleneceği tasarlanan sistem konumu
DESTDIR="$HOME/distro/rootfs"
# Derleme konumu
ROOTBUILDDIR="/tmp/kly/build"
# Derleme yapılan paketin derleme konumun
BUILDDIR="/tmp/kly/build/build-${name}-${version}"
# paketin derleme talimatının verildiği konum
PACKAGEDIR=$(pwd)
# Paketin kaynak kodlarının olduğu konum
SOURCEDIR="/tmp/kly/build/${name}-${version}"

# initsetup
# derleme dizini yoksa oluşturuluyor
mkdir -p  $ROOTBUILDDIR
# içeriği temizleniyor
rm -rf $ROOTBUILDDIR/*
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 $BUILDDIR

# setup
./configure --prefix=/usr --libdir=/lib64/ --with-included-popt --with-included-zlib --disable-xxhash --disable-lz4

# build
make

# package
make install DESTDIR=$DESTDIR
```

Paket adında(rsync) istediğiniz bir konumda bir dizin oluşturun ve dizin içine giriniz. Yukarı verilen script kodlarını build adında bir dosya oluşturup içine kopyalayın ve kaydedin. Daha sonra build scriptini çalıştırın. Nasıl çalıştırılacağı aşağıdaki komutlarla gösterilmiştir. Aşağıda gösterilen komutları paket için oluşturulan dizinin içinde terminal açarak çalıştırınız.

```
chmod 755 build
fakeroot ./build
```

kbd

KBD paketi, Linux tabanlı sistemlerde klavye ile etkileşimi yönetmek için kritik bir bileşendir. Bu paket, farklı klavye düzenlerini destekler ve kullanıcıların ihtiyaçlarına göre özelleştirilmiş tuş atamaları yapmalarına olanak tanır. Örneğin, bir kullanıcı farklı bir dilde yazmak istediğinde, KBD paketi sayesinde o dilin klavye düzenine geçiş yapabilir.

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"
group="sys.apps"

# Paketin yükleneceği tasarlanan sistem konumu
DESTDIR="$HOME/distro/rootfs"
# Derleme konumu
ROOTBUILDDIR="/tmp/kly/build"
# Derleme yapılan paketin derleme konumun
BUILDDIR="/tmp/kly/build/build-${name}-${version}"
# paketin derleme talimatının verildiği konum
PACKAGEDIR=$(pwd)
# Paketin kaynak kodlarının olduğu konum
SOURCEDIR="/tmp/kly/build/${name}-${version}"

# initsetup
# derleme dizini yoksa oluşturuluyor
mkdir -p  $ROOTBUILDDIR
# içeriği temizleniyor
rm -rf $ROOTBUILDDIR/*
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 $BUILDDIR

# 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
```

Yukarıdaki kodların sorunsuz çalışabilmesi için ek dosyayalara ihtiyaç vardır. Bu ek dosyaları indirmek için tıklayınız.

tar dosyasını indirdikten sonra istediğiniz bir konumda **kbd** adında bir dizin oluşturun ve tar dosyasını oluşturulan dizin içinde açınınız.

Paket adında(kbd) istediğiniz bir konumda bir dizin oluşturun ve dizin içine giriniz. Yukarı verilen script kodlarını build adında bir dosya oluşturup içine kopyalayın ve kaydedin. Daha sonra build scriptini çalıştırın. Nasıl çalıştırılacağı aşağıdaki komutlarla gösterilmiştir. Aşağıda gösterilen komutları paket için oluşturulan dizinin içinde terminal açarak çalıştırınız.

```
chmod 755 build
fakeroot ./build
```

kernel

Kernel, bilgisayar sistemlerinde işletim sisteminin kalbini oluşturan bir yazılım katmanıdır. Donanım kaynaklarını yönetir, sistem çağrılarını işler ve uygulama yazılımlarının donanım ile etkileşimini sağlar. Linux işletim sisteminde, kernel, çoklu görev yönetimi, bellek yönetimi, dosya sistemi erişimi ve ağ iletişimi gibi kritik işlevleri yerine getirir.

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 kernelini indirip kendi sistemimize yükleyen bir işlem yapacağız. Fakat derlemek isterseniz Derleme başlığı altında paylaşılan scripti kullanabilirsiniz. Kerneli hazırladığımız sistemem kurmak için aşağıda script verilmiştir.

Debian Kernel

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

# Paketin yükleneceği tasarlanan sistem konumu
DESTDIR="$HOME/distro/rootfs"
# Derleme konumu
ROOTBUILDDIR="/tmp/kly/build"
# Derleme yapılan paketin derleme konumun
BUILDDIR="/tmp/kly/build/build-${name}-${version}"
# paketin derleme talimatının verildiği konum
PACKAGEDIR=$(pwd)
# Paketin kaynak kodlarının olduğu konum
SOURCEDIR="/tmp/kly/build/${name}-${version}"

# initsetup
# derleme dizini yoksa oluşturuluyor
mkdir -p  $ROOTBUILDDIR
# içeriği temizleniyor
rm -rf $ROOTBUILDDIR/*
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 $BUILDDIR

# 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

# package
cd $SOURCEDIR
cp -prfv boot  ${DESTDIR}/
cp -prfv lib/*  ${DESTDIR}/lib/
find ${DESTDIR}/ -iname "*" -exec unxz {} \;
```

Kernel 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-$version.tar.xz"
depends="kernel"
builddepend="rsync,bc,cpio,gettext,elfutils,pahole,perl,python,tar,xz-utils"
group="sys.kernel"
# Paketin yükleneceği tasarlanan sistem konumu
DESTDIR="$HOME/distro/rootfs"
# Derleme konumu
ROOTBUILDDIR="/tmp/kly/build"
# Derleme yapılan paketin derleme konumun
BUILDDIR="/tmp/kly/build/build-${name}-${version}"
# paketin derleme talimatının verildiği konum
PACKAGEDIR=$(pwd)
# Paketin kaynak kodlarının olduğu konum
SOURCEDIR="/tmp/kly/build/${name}-${version}"

# initsetup
# derleme dizini yoksa oluşturuluyor
mkdir -p  $ROOTBUILDDIR
# içeriği temizleniyor
rm -rf $ROOTBUILDDIR/*
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 $BUILDDIR

# setup
cp -prvf $PACKAGEDIR/files/ $SOURCEDIR/
patch -Np1 -i $PACKAGEDIR/files/patch-$version
cp $PACKAGEDIR/files/config $SOURCEDIR/.config
make olddefconfig

# build
make bzImage -j$(nproc)
make modules -j$(nproc)
# package
#----------------------------------------                                        install                         ------------------------------------
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=$DESTDIR/usr
#----------------------------------------                                  install headers                   ------------------------------------
mkdir -p "$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 "$kernelbuilddir" -type f -name '*.o' -printf 'Removing %P\n' -delete
#----------------------------------------                                  install                                                                                                 ------------------------------------
if [[ -d "$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
```

Yukarıdaki kodların sorunsuz çalışabilmesi için ek dosyayalara ihtiyaç vardır. Bu ek dosyaları indirmek için tıklayınız. tar dosyasını indirdikten sonra istediğiniz bir konumda **kernel** adında bir dizin oluşturun ve tar dosyasını oluşturulan dizin içinde açınınız. Yukarı verilen script kodlarını build adında bir dosya oluşturup içine kopyalayın ve kaydedin. Daha sonra build scriptini çalıştırın. Aşağıda gösterilen komutları paket için oluşturulan dizinin içinde terminal açarak çalıştırınız.

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

dialog

Dialog paketi, terminal tabanlı uygulamalar için kullanıcı ile etkileşim kurmayı kolaylaştıran bir araçtır. Bu paket, kullanıcıdan bilgi almak veya kullanıcıya bilgi sunmak amacıyla çeşitli diyalog kutuları oluşturmanıza olanak tanır. Örneğin, metin kutuları, onay kutuları, seçim kutuları gibi farklı türde diyaloglar oluşturabilirsiniz.

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"
groups="sys.apps"

# Paketin yükleneceği tasarlanan sistem konumu
DESTDIR="$HOME/distro/rootfs"
# Derleme konumu
ROOTBUILDDIR="/tmp/kly/build"
# Derleme yapılan paketin derleme konumun
BUILDDIR="/tmp/kly/build/build-${name}-${version}"
# paketin derleme talimatının verildiği konum
PACKAGEDIR=$(pwd)
# Paketin kaynak kodlarının olduğu konum
SOURCEDIR="/tmp/kly/build/${name}-${version}"

# initsetup
# derleme dizini yoksa oluşturuluyor
mkdir -p  $ROOTBUILDDIR
# içeriği temizleniyor
rm -rf $ROOTBUILDDIR/*
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 $BUILDDIR

# setup
./configure --prefix=/usr --libdir=/lib64/ --with-ncursesw

# build
make

# package
make install DESTDIR=$DESTDIR
```

Paket adında(dialog) istediğiniz bir konumda bir dizin oluşturun ve dizin içine giriniz. Yukarı verilen script kodlarını build adında bir dosya oluşturup içine kopyalayın ve kaydedin. Daha sonra build scriptini çalıştırın. Nasıl çalıştırılacağı aşağıdaki komutlarla gösterilmiştir. Aşağıda gösterilen komutları paket için oluşturulan dizinin içinde terminal açarak çalıştırınız.

```
chmod 755 build
fakeroot ./build
```

live-boot

Live-boot paketi, kullanıcıların bir işletim sistemini kurmadan önce denemelerine olanak tanıyan bir araçtır. Genellikle bir USB bellek veya CD/DVD gibi taşınabilir bir ortamda bulunur. Bu paket, sistemin kurulu olduğu ortamdan bağımsız olarak çalışır ve kullanıcıların işletim sisteminin özelliklerini, performansını ve uyumluluğunu test etmelerine imkan tanır.

Derleme

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/live-boot/-/archive/debian/1%2520230131/live-boot-debian-1%2520230131.tar.gz"
groups="sys.kernel"

# Paketin yükleneceği tasarlanan sistem konumu
DESTDIR="$HOME/distro/rootfs"
# Derleme konumu
ROOTBUILDDIR="/tmp/kly/build"
# Derleme yapılan paketin derleme konumun
BUILDDIR="/tmp/kly/build/build-${name}-${version}"
# paketin derleme talimatının verildiği konum
PACKAGEDIR=$(pwd)
# Paketin kaynak kodlarının olduğu konum
SOURCEDIR="/tmp/kly/build/${name}-${version}"

# initsetup
# derleme dizini yoksa oluşturuluyor
mkdir -p  $ROOTBUILDDIR
# içeriği temizleniyor
rm -rf $ROOTBUILDDIR/*
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 $BUILDDIR

# setup

# 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
```

Paket adında(live-boot) istediğiniz bir konumda bir dizin oluşturun ve dizin içine giriniz. Yukarı verilen script kodlarını build adında bir dosya oluşturup içine kopyalayın ve kaydedin. Daha sonra build scriptini çalıştırın. Nasıl çalıştırılacağı aşağıdaki komutlarla gösterilmiştir. Aşağıda gösterilen komutları paket için oluşturulan dizinin içinde terminal açarak çalıştırınız.

```
chmod 755 build
fakeroot ./build
```

live-config

Live-Config, özellikle canlı CD veya USB ortamlarında kullanılan bir yapılandırma paketidir. Bu paket, sistemin başlangıçta otomatik olarak belirli ayarlarla başlatılmasını sağlar. Örneğin, ağ ayarları, kullanıcı hesapları ve diğer sistem yapılandırmaları gibi unsurlar, Live-Config aracılığıyla önceden tanımlanabilir.

Kullanıcılar, Live-Config ile özelleştirilmiş bir canlı sistem oluşturabilir ve bu sistemin her açılışında belirli ayarların otomatik olarak uygulanmasını sağlayabilir. Bu, özellikle eğitim, test veya kurtarma senaryolarında büyük bir avantaj sunar.

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"
groups="sys.kernel"

# Paketin yükleneceği tasarlanan sistem konumu
DESTDIR="$HOME/distro/rootfs"
# Derleme konumu
ROOTBUILDDIR="/tmp/kly/build"
# Derleme yapılan paketin derleme konumun
BUILDDIR="/tmp/kly/build/build-${name}-${version}"
# paketin derleme talimatının verildiği konum
PACKAGEDIR=$(pwd)
# Paketin kaynak kodlarının olduğu konum
SOURCEDIR="/tmp/kly/build/${name}-${version}"

# initsetup
# derleme dizini yoksa oluşturuluyor
mkdir -p  $ROOTBUILDDIR
# içeriği temizleniyor
rm -rf $ROOTBUILDDIR/*
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 $BUILDDIR

# setup

# build
make

# package
make install DESTDIR=$DESTDIR
```

Paket adında(live-config) istediğiniz bir konumda bir dizin oluşturun ve dizin içine giriniz. Yukarı verilen script kodlarını build adında bir dosya oluşturup içine kopyalayın ve kaydedin. Daha sonra build scriptini çalıştırın. Nasıl çalıştırılacağı aşağıdaki komutlarla gösterilmiştir. Aşağıda gösterilen komutları paket için oluşturulan dizinin içinde terminal açarak çalıştırınız.

```
chmod 755 build
fakeroot ./build
```

parted

Parted, Linux tabanlı sistemlerde disk bölümlerini oluşturmak, silmek, boyutlandırmak ve düzenlemek için kullanılan bir komut satırı aracıdır. Kullanıcıların disk alanlarını daha verimli bir şekilde yönetmelerine yardımcı olur. Parted, hem MBR (Master Boot Record) hem de GPT (GUID Partition Table) bölümlendirme şemalarını destekler.

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"
groups="sys.block"

# Paketin yükleneceği tasarlanan sistem konumu
DESTDIR="$HOME/distro/rootfs"
# Derleme konumu
ROOTBUILDDIR="/tmp/kly/build"
# Derleme yapılan paketin derleme konumun
BUILDDIR="/tmp/kly/build/build-${name}-${version}"
# paketin derleme talimatının verildiği konum
PACKAGEDIR=$(pwd)
# Paketin kaynak kodlarının olduğu konum
SOURCEDIR="/tmp/kly/build/${name}-${version}"

# initsetup
# derleme dizini yoksa oluşturuluyor
mkdir -p  $ROOTBUILDDIR
# içeriği temizleniyor
rm -rf $ROOTBUILDDIR/*
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 $BUILDDIR

# setup
./configure --prefix=/usr --libdir=/usr/lib64/ --sbindir=/usr/bin --disable-rpath --disable-device-mapper

# build
make

# package
make install DESTDIR=$DESTDIR
```

Paket adında(parted) istediğiniz bir konumda bir dizin oluşturun ve dizin içine giriniz. Yukarı verilen script kodlarını build adında bir dosya oluşturup içine kopyalayın ve kaydedin. Daha sonra build scriptini çalıştırın. Nasıl çalıştırılacağı aşağıdaki komutlarla gösterilmiştir. Aşağıda gösterilen komutları paket için oluşturulan dizinin içinde terminal açarak çalıştırınız.

```
chmod 755 build
fakeroot ./build
```

busybox

BusyBox, Linux tabanlı sistemlerde "İsviçre Çakısı" benzeri bir işlevsellik sunarak, çeşitli komutları tek bir ikili dosya altında toplar.

BusyBox, ls, cp, mv, rm gibi temel komutların yanı sıra, ağ yönetimi, dosya sistemleri ve sistem yönetimi gibi birçok alanda işlevsellik sunar. Kullanıcılar, bu komutları BusyBox ile çağırarak 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="minimal linux araç paketi static derlenmiş hali"
source="https://busybox.net/downloads/${name}-${version}.tar.bz2"
group="sys.base"

# Paketin yükleneceği tasarlanan sistem konumu
DESTDIR="$HOME/distro/rootfs"
# Derleme konumu
ROOTBUILDDIR="/tmp/kly/build"
# Derleme yapılan paketin derleme konumun
BUILDDIR="/tmp/kly/build/build-${name}-${version}"
# paketin derleme talimatının verildiği konum
PACKAGEDIR=$(pwd)
# Paketin kaynak kodlarının olduğu konum
SOURCEDIR="/tmp/kly/build/${name}-${version}"

# initsetup
# derleme dizini yoksa oluşturuluyor
mkdir -p  $ROOTBUILDDIR
# içeriği temizleniyor
rm -rf $ROOTBUILDDIR/*
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 $BUILDDIR

# 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
mkdir -p ${DESTDIR}/usr/share/udhcpc/ ${DESTDIR}/etc/init.d/
install $SOURCEDIR/files/udhcpc.script ${DESTDIR}/usr/share/udhcpc/default.script              # install udhcpc script and service
install $SOURCEDIR/files/udhcpc.openrc ${DESTDIR}/etc/init.d/udhcpc
cd $DESTDIR/bin&&ln -s busybox hostname
```

Yukarıdaki kodların sorunsuz çalışabilmesi için ek dosyayalara ihtiyaç vardır. Bu ek dosyaları indirmek için tıklayınız.

tar dosyasını indirdikten sonra istediğiniz bir konumda **busybox** adında bir dizin oluşturun ve tar dosyasını oluşturulan dizin içinde açınınız.

Paket adında(busybox) istediğiniz bir konumda bir dizin oluşturun ve dizin içine giriniz. Yukarı verilen script kodlarını build adında bir dosya oluşturup içine kopyalayın ve kaydedin. Daha sonra build scriptini çalıştırın. Nasıl çalıştırılacağı aşağıdaki komutlarla gösterilmiştir. Aşağıda gösterilen komutları paket için oluşturulan dizinin içinde terminal açarak çalıştırınız.

```
chmod 755 build
fakeroot ./build
```

nano

Nano, terminal tabanlı bir metin düzenleyici olup, GNU projesinin bir parçasıdır. Kullanıcıların metin dosyalarını kolayca oluşturmasına, düzenlemesine ve kaydetmesine olanak tanır. Nano, vi veya emacs gibi daha karmaşık metin düzenleyicilere göre daha basit bir kullanım sunar.

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"
groups="app.editor"

# Paketin yükleneceği tasarlanan sistem konumu
DESTDIR="$HOME/distro/rootfs"
# Derleme konumu
ROOTBUILDDIR="/tmp/kly/build"
# Derleme yapılan paketin derleme konumun
BUILDDIR="/tmp/kly/build/build-${name}-${version}"
# paketin derleme talimatının verildiği konum
PACKAGEDIR=$(pwd)
# Paketin kaynak kodlarının olduğu konum
SOURCEDIR="/tmp/kly/build/${name}-${version}"

# initsetup
# derleme dizini yoksa oluşturuluyor
mkdir -p  $ROOTBUILDDIR
# içeriği temizleniyor
rm -rf $ROOTBUILDDIR/*
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 $BUILDDIR

# setup
./configure --prefix=/usr

# build
make

# package
make install DESTDIR=$DESTDIR
cd $DESTDIR
mkdir -p $DESTDIR/lib
echo "INPUT(-lncursesw)" > $DESTDIR/lib/libncurses.so
```

Paket adında(nano) istediğiniz bir konumda bir dizin oluşturun ve dizin içine giriniz. Yukarı verilen script kodlarını build adında bir dosya oluşturup içine kopyalayın ve kaydedin. Daha sonra build scriptini çalıştırın. Nasıl çalıştırılacağı aşağıdaki komutlarla gösterilmiştir. Aşağıda gösterilen komutları paket için oluşturulan dizinin içinde terminal açarak çalıştırınız.

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

grub

GRUB (GRand Unified Bootloader), çoklu işletim sistemlerini destekleyen ve kullanıcıların sistemlerini başlatmalarını sağlayan bir önyükleyici yazılımıdır. 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
        }

# Paketin yükleneceği tasarlanan sistem konumu
DESTDIR="$HOME/distro/rootfs"
# Derleme konumu
ROOTBUILDDIR="/tmp/kly/build"
# Derleme yapılan paketin derleme konumun
BUILDDIR="/tmp/kly/build/build-${name}-${version}"
# paketin derleme talimatının verildiği konum
PACKAGEDIR=$(pwd)
# Paketin kaynak kodlarının olduğu konum
SOURCEDIR="/tmp/kly/build/${name}-${version}"

# initsetup
# derleme dizini yoksa oluşturuluyor
mkdir -p  $ROOTBUILDDIR
# içeriği temizleniyor
rm -rf $ROOTBUILDDIR/*
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 $BUILDDIR

# 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 $jobs -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
```

Yukarıdaki kodların sorunsuz çalışabilmesi için ek dosyayalara ihtiyaç vardır. Bu ek dosyaları indirmek için tıklayınız.

tar dosyasını indirdikten sonra istediğiniz bir konumda **grub** adında bir dizin oluşturun ve tar dosyasını oluşturulan dizin içinde açınınız. Oluşturuluna dizinde yukarı verilen script kodlarını build adında bir dosya oluşturup içine kopyalayın ve kaydedin. Daha sonra build scriptini çalıştırın. Nasıl çalıştırılacağı aşağıdaki komutlarla gösterilmiştir. Aşağıda gösterilen komutları paket için oluşturulan dizinin içinde terminal açarak çalıştırınız.

```
chmod 755 build
fakeroot ./build
```

efibootmgr

efibootmgr, UEFI (Unified Extensible Firmware Interface) tabanlı sistemlerde önyükleme yöneticisi olarak işlev gören bir Linux aracıdır. Bu paket, UEFI önyükleme seçeneklerini yönetmek için kullanılır ve sistemin önyükleme sırasını, önyükleme girişlerini ve diğer ilgili ayarları düzenlemeye olanak tanır.

Derleme

Debian ortamında bu paketin derlenmesi için;

- **sudo apt install libefiboot-dev**
- **sudo cp -prfv /usr/include/efivar/* /usr/include/**

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=""
group="sys.boot"

# Paketin yükleneceği tasarlanan sistem konumu
DESTDIR="$HOME/distro/rootfs"
# Derleme konumu
ROOTBUILDDIR="/tmp/kly/build"
# Derleme yapılan paketin derleme konumun
BUILDDIR="/tmp/kly/build/build-${name}-${version}"
# paketin derleme talimatının verildiği konum
PACKAGEDIR=$(pwd)
# Paketin kaynak kodlarının olduğu konum
SOURCEDIR="/tmp/kly/build/${name}-${version}"

# initsetup
# derleme dizini yoksa oluşturuluyor
mkdir -p  $ROOTBUILDDIR
# içeriği temizleniyor
rm -rf $ROOTBUILDDIR/*
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 $BUILDDIR

# setup
# build
make sbindir=/usr/bin EFIDIR=/boot/efi PCDIR=/usr/lib64/pkgconfig

# package
EFIDIR="/boot/efi" sbindir=/usr/bin make DESTDIR="$DESTDIR" install
```

Paket adında(efibootmgr) istediğiniz bir konumda bir dizin oluşturun ve dizin içine giriniz. Yukarı verilen script kodlarını build adında bir dosya oluşturup içine kopyalayın ve kaydedin. Daha sonra build scriptini çalıştırın. Nasıl çalıştırılacağı aşağıdaki komutlarla gösterilmiştir. Aşağıda gösterilen komutları paket için oluşturulan dizinin içinde terminal açarak çalıştırınız.

```
chmod 755 build
fakeroot ./build
```

efivar

efivar paketi, UEFI tabanlı sistemlerde, firmware ile işletim sistemi arasında veri alışverişini sağlamak için kritik bir rol oynamaktadır. UEFI, BIOS'un yerini alarak daha modern bir arayüz sunmakta ve sistem başlangıcında daha fazla esneklik sağlamaktadır. efivar, UEFI değişkenlerini okuma, yazma ve silme işlemlerini gerçekleştirmek için bir dizi komut sunar.

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=""
group="sys.libs"

# Paketin yükleneceği tasarlanan sistem konumu
DESTDIR="$HOME/distro/rootfs"
# Derleme konumu
ROOTBUILDDIR="/tmp/kly/build"
# Derleme yapılan paketin derleme konumun
BUILDDIR="/tmp/kly/build/build-${name}-${version}"
# paketin derleme talimatının verildiği konum
PACKAGEDIR=$(pwd)
# Paketin kaynak kodlarının olduğu konum
SOURCEDIR="/tmp/kly/build/${name}-${version}"

# initsetup
# derleme dizini yoksa oluşturuluyor
mkdir -p  $ROOTBUILDDIR
# içeriği temizleniyor
rm -rf $ROOTBUILDDIR/*
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 $BUILDDIR

# 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
```

Paket adında(efivar) istediğiniz bir konumda bir dizin oluşturun ve dizin içine giriniz. Yukarı verilen script kodlarını build adında bir dosya oluşturup içine kopyalayın ve kaydedin. Daha sonra build scriptini çalıştırın. Nasıl çalıştırılacağı aşağıdaki komutlarla gösterilmiştir. Aşağıda gösterilen komutları paket için oluşturulan dizinin içinde terminal açarak çalıştırınız.

```
chmod 755 build
fakeroot ./build
```

libssh

libssh, SSH protokolünü uygulamak için kullanılan açık kaynaklı bir C kütüphanesidir. Bu kütüphane, geliştiricilere güvenli bir şekilde veri iletimi, uzaktan erişim ve dosya transferi gibi işlevleri gerçekleştirme imkanı tanır. libssh, hem istemci hem de sunucu tarafında kullanılabilir ve çoklu platform desteği sunar.

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")
group="net.libs"
depends="openssl,zlib"

# Paketin yükleneceği tasarlanan sistem konumu
DESTDIR="$HOME/distro/rootfs"
# Derleme konumu
ROOTBUILDDIR="/tmp/kly/build"
# Derleme yapılan paketin derleme konumun
BUILDDIR="/tmp/kly/build/build-${name}-${version}"
# paketin derleme talimatının verildiği konum
PACKAGEDIR=$(pwd)
# Paketin kaynak kodlarının olduğu konum
SOURCEDIR="/tmp/kly/build/${name}-${version}"

# initsetup
# derleme dizini yoksa oluşturuluyor
mkdir -p  $ROOTBUILDDIR
# içeriği temizleniyor
rm -rf $ROOTBUILDDIR/*
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 $BUILDDIR

# 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/
```

Paket adında(libssh) istediğiniz bir konumda bir dizin oluşturun ve dizin içine giriniz. Yukarı verilen script kodlarını build adında bir dosya oluşturup içine kopyalayın ve kaydedin. Daha sonra build scriptini çalıştırın. Nasıl çalıştırılacağı aşağıdaki komutlarla gösterilmiştir. Aşağıda gösterilen komutları paket için oluşturulan dizinin içinde terminal açarak çalıştırınız.

```
chmod 755 build
fakeroot ./build
```

openssh

OpenSSH, Secure Shell (SSH) protokolünü uygulayan bir yazılım paketidir ve genellikle Linux ve Unix tabanlı sistemlerde kullanılır. Bu paket, kullanıcıların uzak sunuculara güvenli bir şekilde bağlanmalarını, dosya transferi yapmalarını ve uzaktan komut çalıştırmalarını sağlar. OpenSSH, veri iletimini şifreleyerek, ağ üzerinden yapılan iletişimlerin güvenliğini artırır.

OpenSSH, genellikle ssh, scp, sftp ve sshd gibi araçları içerir.

Derleme

Debian ortamında bu paketin derlenmesi için; **sudo apt install libcrypt-dev** komutuyla paketin 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"
group="net.misc"

# Paketin yükleneceği tasarlanan sistem konumu
DESTDIR="$HOME/distro/rootfs"
# Derleme konumu
ROOTBUILDDIR="/tmp/kly/build"
# Derleme yapılan paketin derleme konumun
BUILDDIR="/tmp/kly/build/build-${name}-${version}"
# paketin derleme talimatının verildiği konum
PACKAGEDIR=$(pwd)
# Paketin kaynak kodlarının olduğu konum
SOURCEDIR="/tmp/kly/build/${name}-${version}"

# initsetup
# derleme dizini yoksa oluşturuluyor
mkdir -p  $ROOTBUILDDIR
# içeriği temizleniyor
rm -rf $ROOTBUILDDIR/*
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 $BUILDDIR

# 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,sysconf,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
```

```
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" /etc/ssh/sshd_config
echo -e "\nPermitRootLogin yes">> /etc/ssh/sshd_config
```

Yukarıdaki kodların sorunsuz çalışabilmesi için ek dosyayalara ihtiyaç vardır. Bu ek dosyaları indirmek için tıklayınız.

tar dosyasını indirdikten sonra istediğiniz bir konumda **openssh** adında bir dizin oluşturun ve tar dosyasını oluşturulan dizin içinde açınınız.

Paket adında(openssh) istediğiniz bir konumda bir dizin oluşturun ve dizin içine giriniz. Yukarı verilen script kodlarını build adında bir dosya oluşturup içine kopyalayın ve kaydedin. Daha sonra build scriptini çalıştırın. Nasıl çalıştırılacağı aşağıdaki komutlarla gösterilmiştir. Aşağıda gösterilen komutları paket için oluşturulan dizinin içinde terminal açarak çalıştırınız.

```
chmod 755 build
fakeroot ./build
```

pam

PAM, "Pluggable Authentication Modules" ifadesinin kısaltmasıdır ve Linux işletim sistemlerinde kimlik doğrulama süreçlerini esnek bir şekilde yönetmek için tasarlanmıştır. PAM, sistem yöneticilerine, kimlik doğrulama yöntemlerini modüler bir yapıda değiştirme ve özelleştirme imkanı sunar. Örneğin, bir sistem yöneticisi, kullanıcıların şifre ile kimlik doğrulamasını sağlarken, aynı zamanda iki faktörlü kimlik doğrulama gibi ek güvenlik önlemleri de ekleyebilir.

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"
groups="sys.libs"

# Paketin yükleneceği tasarlanan sistem konumu
DESTDIR="$HOME/distro/rootfs"
# Derleme konumu
ROOTBUILDDIR="/tmp/kly/build"
# Derleme yapılan paketin derleme konumun
BUILDDIR="/tmp/kly/build/build-${name}-${version}"
# paketin derleme talimatının verildiği konum
PACKAGEDIR=$(pwd)
# Paketin kaynak kodlarının olduğu konum
SOURCEDIR="/tmp/kly/build/${name}-${version}"

# initsetup
# derleme dizini yoksa oluşturuluyor
mkdir -p  $ROOTBUILDDIR
# içeriği temizleniyor
rm -rf $ROOTBUILDDIR/*
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 $BUILDDIR

# 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
```

Paket adında(pam) istediğiniz bir konumda bir dizin oluşturun ve dizin içine giriniz. Yukarı verilen script kodlarını build adında bir dosya oluşturup içine kopyalayın ve kaydedin. Daha sonra build scriptini çalıştırın. Nasıl çalıştırılacağı aşağıdaki komutlarla gösterilmiştir. Aşağıda gösterilen komutları paket için oluşturulan dizinin içinde terminal açarak çalıştırınız.

```
chmod 755 build
fakeroot ./build
```

Paket Sistemi

Paket Sitemi

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

Paket sistemeleri bir paketi yüklemek istediğinde, genellikle daha önceden derlenmiş paketleri veya yükeleme 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 sistemeleri şunlardır.

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

kly Paket Sistemi

Bu dokümanda hazırlanan dağıtımın paket sistemi için ise kly(KendiLinuxunuYap=kolay) 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.

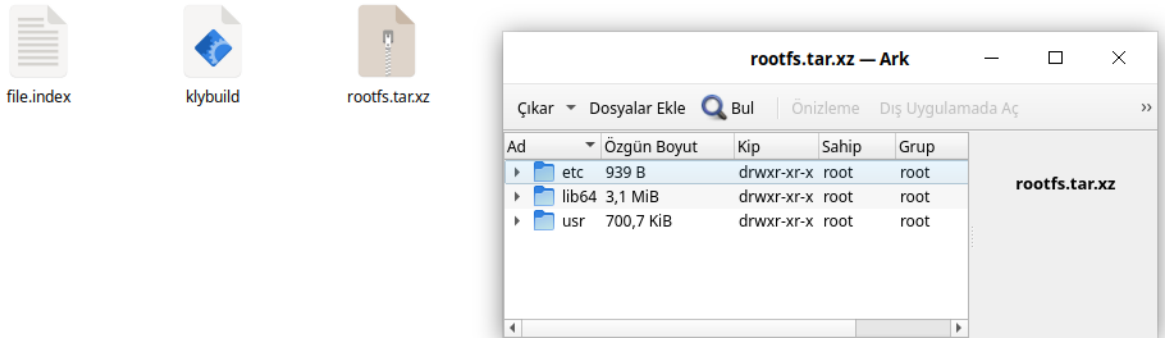
Paket Oluşturma

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.

Daha önceden derlediğimiz readline paketinin scriptini, aşağıdaki gibi düzenlendi.

```
#!/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"

# Paketin yükleneceği tasarlanan sistem konumu
DESTDIR="$HOME/distro/rootfs"
# Derleme konumu
ROOTBUILDDIR="/tmp/kly/build"
# Derleme yapılan paketin derleme konumun
BUILDDIR="/tmp/kly/build/build-${name}-${version}"
# paketin derleme talimatının verildiği konum
PACKAGEDIR=$(pwd)
# Paketin kaynak kodlarının olduğu konum
SOURCEDIR="/tmp/kly/build/${name}-${version}"

initsetup(){
        # derleme dizini yoksa oluşturuluyor
        mkdir -p  $ROOTBUILDDIR
        # içeriği temizleniyor
        rm -rf $ROOTBUILDDIR/*
        # dizinine geçiyoruz
        cd $ROOTBUILDDIR
        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 $BUILDDIR
}

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

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 ${paket}/klybuild
# Paketin yükleneceği tasarlanan sistem konumu
DESTDIR="$HOME/distro/rootfs"
# Derleme konumu
ROOTBUILDDIR="/tmp/kly/build"
# Derleme yapılan paketin derleme konumun
BUILDDIR="/tmp/kly/build/build-${name}-${version}"
# paketin derleme talimatının verildiği konum
PACKAGEDIR=$(pwd)
# Paketin kaynak kodlarının olduğu konum
SOURCEDIR="/tmp/kly/build/${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**

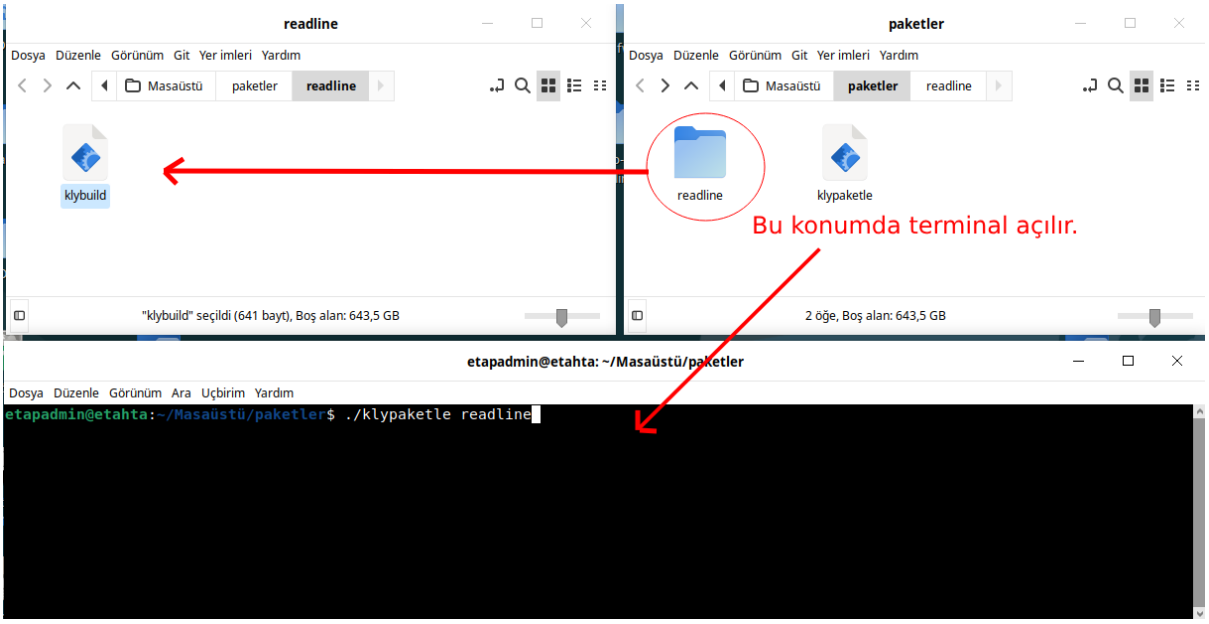
dosyası hazırlayacağız. **klypaketle** dosyamızda değişiklik yapmayacağız. Artık **klypaketle** dosyası paketimizi oluşturan script **klybuild** ise hazırlanacak paketin bilgilerini bulunduran script doyasıdır.

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. Paketler genel olarak;

1. Sıkıştırılmış bir dosyadır
2. İçerisinde sisteme yüklenecek derlenmiş paket dizini(sıkıştırılmış olur genelde)
3. Sisteme yüklenecek derlenmiş paket dizini içindeki dosyaların ve dizinleri konumu ve listesi(file.lst)
4. Paket derleme talimatı.

Depoada 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** dosyasından alınır. Paketlerin listesi oluşturlduktan sonra paketler kurulurken, silinirken ve güncellenirken bu listeden yararlanılır.

```
paket_adi: bash
paket_versiyonu: 1.0
paket_bagimliliklari: glibc, ncurses, readline
paket_konumu: b/bash/bash_1.0.zip

paket_adi: test
paket_versiyonu: 1.1
paket_bagimliliklari:
paket_konumu: t/test/test_2.0.zip
```

Yukarıdaki örnekte paket adı bilgisi sürüm bilgisi ve bağımılılıklar gibi bilgiler ile paketin sunucu içerisindeki konumu yer almaktadır. Depo indexi paketlerin içinde yer alan paket bilgileri okunarak otomatik olarak oluşturulur.

Örneğin paketlerimiz zip dosyası olsun ve paket bilgisini **build** dosyası taşısın. Aşağıdaki gibi depo indexi alabiliriz.

```
function index {
    > index.txt
    for i in $@ ; do
        unzip -p $i build >> index.txt
        echo "Paket_Konumu: $i" >> index.txt
    done
}

index t/test/test_2.0.zip b/bash/bash_1.0.zip
```

Bu örnekte paketlerin içindeki paket bilgisi içeren dosyaları uç uca ekledik. Buna ek olarak paketin nerede olduğunu anlamak için paket konumunu da ekledik. Çıktısı aşağıdaki gibidir.

```
Paket_Konumu: t/test/test_2.0.zip
Paket_Konumu: b/bash/bash_1.0.zip
```

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 ouşturarak aşağıda verilen main.yml dosyasını oluşturunuz.
- internet üzerinden kly-binary-packages reposunda settings->action->general->Workflow permissions->Read and write permissions işaretlenmelidir.
- Yapılan paketler github üzerinde gönderilmelidir.

index Dosyası

Bu script kly paket dosyalarımızın olduğu dizinde tüm paketleri açarak içerisinden **klybuild** dosyalarını çıkartarak paketle ilgili bilgileri alıp **index.lst** dosyası oluşturmaktadır. istersek 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
```

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 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 gerekmetedir. 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 sitemindede /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) yüklü paketlerin index bilgilerini tutan paket sistemi dizininindeki index dosyasına eklenir.
5. Açılan paket içindeki yüklenen dosyaların nereye yüklendiğini tutan file.index dosyası paket sistemi dizinine yüklenir

Bu işlemler daha detaylandırılabilir. Bu işlemlerin detaylı olması paket sisteminin kullanılabilirliğini ve yetenekleri olarak ifade edebiliriz. İşlem adımlarını kolaylıkla sıralarken bunları yapacak script yazmak ciddi planlamalar yapılarak tasarlanması gerekmektedir.

Örneğin bir paketimiz zip dosyası olsun ve içinde dosya listesini tutan **file.index** adında bir dosyamız olsun. Paketi aşağıdaki gibi kurabiliriz.

```
cd /tmp/kur/
unzip /dosya/yolu/paket.zip
cp -rfp ./* /
cp file.index /paket/veri/yolu/paket.index
```

- Bu örnekte ilk satırda geçici dizine gittik
- Paketi oraya açtık.
- Paket içeriğini kök dizine kopyaladık.
- Paket dosya listesini verilerin tutulduğu yere kopyaladık.

Bu işlemden sonra paket kurulmuş oldu.

kly Paket Kurma Scripti Tasarlama

Burada basit seviyede kurulum yapan script kullanılmıştır. Detaylandırıldıkça doküman güncellenecektir. Kurulum scripti aşağıda görülmektedir.

Paket kurulurken paket içerisinde bulunan dosyalar sisteme kopyalanır. Daha sonra istenirse silinebilmesi için paket içeriğinde dosyaların listesi tutulur. Bu dosya ayrıca paketin bütünlüğünü kontrol etmek için de kullanılır.

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 paket.LIST dosyası içerisindeki dosyalar, dizinler kaldırılır.
3. Kaldırılan dosyalardan sonra /paket/veri/yolu/paket.LIST dosyasından paket bilgisi kaldırılır.
4. sistemde kurulu paketler index dosyasından ilgili paket satırı kaldırılmalıdır.

Paketi kaldırmak için ise aşağıdaki örnek kullanılabilir.

```
cat /paket/veri/yolu/paket.LIST | while read dosya ; do
    if [[ -f "$dosya" ]] ; then
        rm -f "$dosya"
    fi
done
cat /paket/veri/yolu/paket.LIST | while read dizin ; do
    if [[ -d "$dizin" ]] ; then
        rmdir "$dizin" || true
    fi
done
rm -f /paket/veri/yolu/paket.LIST
```

Bu örnekte paket listesini satır satır okuduk. Önce dosya olanları sildik. Daha sonra tekrar okuyup boş kalan dizinleri sildik. Son olarak palet listesi dosyamızı sildik. Bu işlem sonunda paket silinmiş oldu.

kly Paket Kaldırma Scripti Tasarlama

Dokumanda örnek olarak verilen kly paket sistemi için yukarıdaki paket kaldırma bilgilerini kullanarak tasarlanmıştır.

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"

# Paketin file.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
# /var/lib/kly/paket-version.lst dosyasından paket bilgisi kaldırılır.
rm -f $target/var/lib/kly/${paket}-${version}.lst

# /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 **********"
```

Bu örnekte paket listesini satır satır okuduk. Önce dosya olanları sildik. Daha sonra tekrar okuyup boş kalan dizinleri sildik. Son olarak paket listesi dosyamızı sildik. Bu işlem sonunda paket silinmiş oldu.

klykaldir Kullanma

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

iso Hazırlama

İso Hazırlama

Önceki bölümlerde paketler derlendi. Bu paketler oturum açtığınız kullanıcı ev dizinde **$HOME/distro/rootfs** konumunda olacaktır. Burada **$HOME** açık olan kullanıcı neyse onun konumunu verecektir. Örneğin kullanıcı adımız **bd** olsun. Bu durumda **$HOME** değeri **/home/bd/** olacaktır. Bu örneğimize göre **$HOME/distro/rootfs** ifadesi aslında **/home/bd/distro/rootfs** dir.

İso hazırlama işlemini yazımızın başında minimal iso yapma anlatılmıştı. Burada da benzer yol izlenecek. Sistemimin yani oluşacak **iso** dosyasının yapısı aşağıdaki gibi olacaktır. Buradaki dört dosyanında nasıl hazırlanacağı ayrı ayrı anlatılacak ve en son hepsini kapsayan iso yapma scriptimizi vereceğiz.

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

grub.cfg Hazırlama

Sistemin açılmasını sağlayan dosyalardan birisi **grub.cfg** dosyasıdır. Genel olarak grub.cfg dosyası aşağıdaki gibi olmalıdır.

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

Bu sistem için grub.cfg dosyası aşağıdaki gibi düzenlendi. Burada dikkat edilmesi gereken menü seçeneklerinde **live** ifadesi kulllanılması. Ayrıca sistemimizin servis yönetici ile başlatılmasını istediğimiz için **linux /boot/vmlinuz boot=live init=/sbin/openrc-init net.ifnames=0 biosdevname=0** sistemi canlı açılmasını sağlıyor.

Kurulum için ise canlı(**live**) açıp kurulum yapmasını sağlayan satırımız **linux /boot/vmlinuz boot=live init=/bin/kur quiet** dır.

```
#### Write grub.cfg
# Timeout for menu
echo -e "set timeout=3\n"> $HOME/distro/iso/boot/grub/grub.cfg

# Default boot entry
echo -e "set default=1\n">> $HOME/distro/iso/boot/grub/grub.cfg

# Menu Colours
echo -e "set menu_color_normal=white/black\n">>$HOME/distro/iso/boot/grub/grub.cfg
echo -e "set menu_color_highlight=white\/blue\n">> $HOME/distro/iso/boot/grub/grub.cfg
echo -e "insmod all_video">> $HOME/distro/iso/boot/grub/grub.cfg
echo -e "terminal_output console">> $HOME/distro/iso/boot/grub/grub.cfg
echo -e "terminal_input console">> $HOME/distro/iso/boot/grub/grub.cfg

echo 'menuentry "Canli(live) GNU/Linux 64-bit" --class liveiso  {' >> $HOME/distro/iso/boot/grub/grub.cfg
echo '    linux /boot/vmlinuz boot=live init=/sbin/openrc-init net.ifnames=0 biosdevname=0' >> $HOME/distro/iso/boot/grub/grub.cfg
echo '    initrd /boot/initrd.img' >> $HOME/distro/iso/boot/grub/grub.cfg
echo '}' >> $HOME/distro/iso/boot/grub/grub.cfg

echo 'menuentry "Kur GNU/Linux 64-bit" --class liveiso  {' >> $HOME/distro/iso/boot/grub/grub.cfg
echo '    linux /boot/vmlinuz boot=live init=/bin/kur quiet' >> $HOME/distro/iso/boot/grub/grub.cfg
echo '    initrd /boot/initrd.img' >> $HOME/distro/iso/boot/grub/grub.cfg
echo '}' >> $HOME/distro/iso/boot/grub/grub.cfg
```

initrd.img Oluşturma/Güncelleme

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

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

Eğer bir dizin içinde bir sisteme initrd oluşturlacaksa, yani chroot ile sisteme erişiliyorsa yukarıdaki komut yeterli olmayacaktır. chroot öncesinde sistemin **dev sys proc run** dizinlerinin bağlanılması gerekmektedir. Dizindeki sistemimizin dizin konumu **/$HOME/distro/rootfs** olsun. Buna göre aşağıda sisteme yukarıdaki komutu çalıştırmadan önce çalıştırılması gereken komutlar aşağıda verilmiştir. Dikkat edilmesi gereken en önemli noktalardan biriside bu komutlar **root** yetkisiyle çalıştırılmalıdır.

```
## ------               initrd.img oluşturma scripti             --------
rootfs="$HOME/distro/rootfs"
distro="$HOME/distro"
## Dizinler Oluşturuluyor
mkdir -p $rootfs/dev
mkdir -p $rootfs/sys
mkdir -p $rootfs/proc
mkdir -p $rootfs/run
mkdir -p $rootfs/tmp
## Dizinler bağlanıyor
mount --bind /dev $rootfs/dev
mount --bind /sys $rootfs/sys
mount --bind /proc $rootfs/proc
mount --bind /run$rootfs/run
mount --bind /tmp $rootfs/tmp

### initrd.img oluşturuluyor/güncelleniyor
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

## Dizin bağlantıları kaldırılıyor
umount -lf -R $rootfs/dev 2>/dev/null
umount -lf -R $rootfs/sys 2>/dev/null
umount -lf -R $rootfs/proc 2>/dev/null
umount -lf -R $rootfs/run 2>/dev/null
umount -lf -R $rootfs/tmp 2>/dev/null
#### initrd kopyalanıyor
cp -pf $rootfs/boot/initrd.img-* $distro/iso/boot/initrd.img
```

vmlinuz Hazırlanması

Kernelimizi iso dizinimize taşıyoruz.

```
rootfs="$HOME/distro/rootfs"
distro="$HOME/distro"
#### Copy kernel
cp -pf $rootfs/boot/vmlinuz-* $distro/iso/boot/vmlinuz
#rm -rf $rootfs/boot #istenir boyut küçültmek için bu komut aktifleştirilebilir.
```

filesystem.squashfs Hazırlama

Sistemi **live** kullanma ve yükleme yapabilmek için yapılan sistemi **squashfs** dosya sıkıştırma yöntemiyle sıkılştırıyoruz. Bu dosyayı **$HOME/distro/iso/live/filesystem.squashfs** konumunda olmalı. Aşağıdaki komutlar dosyayı oluşturup **$HOME/distro//iso/live/filesystem.squashfs** konumuna taşımaktadır.

```
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
```

İso Dosyasının Oluşturulması

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

İsonun Test Edilmesi

isolarımız qemu veya virtualbox ile test edilebilir. Linux ortamında terminalden qemu kullanarak aşıdaki gibi test edebilirsiniz. Sistemde qemu paketinin kurulu olması gerekir.

```
# qemu ile isonun test edilmesi.
qemu-system-x86_64 -cdrom $HOME/distro/distro.iso -m 1G
```

iso Oluşturma Scripti

Sisteme giriş yaptığımız kullanıcının ev dizinindeki **distro/rootfs** disininden **distro/iso** dizinini kullanarak **distro** dizinine **distro.iso** dosyasını oluşturan scriptimiz aşağıdadır.

```
#!/bin/bash
# Detect the name of the display in use
display=":$(ls /tmp/.X11-unix/* | sed 's#/tmp/.X11-unix/X##' | head -n 1)"
# Detect the user using such display
user=$(who | grep '('$display')' | awk '{print $1}')

distro="/home/$user/distro"
rootfs="/home/$user/distro/rootfs"
rm -rf "$distro/iso"
### system chroot  bind/mount
for dir in dev dev/pts proc sys; do mount -o bind /$dir $rootfs/$dir; done

chroot $rootfs echo -e "1\n1\n"|chroot $rootfs passwd root
chroot $rootfs useradd live -m -s /bin/sh  -d /home/live
chroot $rootfs echo -e "live\nlive\n"|chroot $rootfs passwd live

for grp in users tty wheel cdrom audio dip video plugdev netdev; do
        chroot $rootfs usermod -aG $grp live || true
done

sed -i "/agetty_options/d" $rootfs/etc/conf.d/agetty
echo -e "\nagetty_options=\"-l /usr/bin/login\"" >> $rootfs/etc/conf.d/agetty


### 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

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

#### system chroot umount
for dir in dev dev/pts proc sys ; do    while umount -lf -R $rootfs/$dir 2>/dev/null ; do true; done done

#***********************iso ********************************
mkdir -p $distro/iso
mkdir -p $distro/iso/boot
mkdir -p $distro/iso/boot/grub
mkdir -p $distro/iso/live || true

#### Copy initramfs
cp -pf $rootfs/boot/initrd.img-* $distro/iso/boot/initrd.img

#### Copy kernel
cp -pf $rootfs/boot/vmlinuz-* $distro/iso/boot/vmlinuz
#rm -rf $rootfs/boot

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

#### Write grub.cfg
# Timeout for menu
echo -e "set timeout=3\n"> $distro/iso/boot/grub/grub.cfg


# Default boot entry
echo -e "set default=1\n">> $distro/iso/boot/grub/grub.cfg

# Menu Colours
echo -e "set menu_color_normal=white/black\n">> $distro/iso/boot/grub/grub.cfg
echo -e "set menu_color_highlight=white\/blue\n">> $distro/iso/boot/grub/grub.cfg
echo -e "insmod all_video">> $distro/iso/boot/grub/grub.cfg
echo -e "terminal_output console">> $distro/iso/boot/grub/grub.cfg
echo -e "terminal_input console">> $distro/iso/boot/grub/grub.cfg

echo 'menuentry "Canli(live) GNU/Linux 64-bit" --class liveiso  {' >> $distro/iso/boot/grub/grub.cfg
echo '    linux /boot/vmlinuz boot=live init=/sbin/openrc-init net.ifnames=0 biosdevname=0' >> $distro/iso/boot/grub/grub.cfg
echo '    initrd /boot/initrd.img' >> $distro/iso/boot/grub/grub.cfg
echo '}' >> $distro/iso/boot/grub/grub.cfg

echo 'menuentry "Kur GNU/Linux 64-bit" --class liveiso  {' >> $distro/iso/boot/grub/grub.cfg
echo '    linux /boot/vmlinuz boot=live init=/bin/kur quiet' >> $distro/iso/boot/grub/grub.cfg
echo '    initrd /boot/initrd.img' >> $distro/iso/boot/grub/grub.cfg
echo '}' >> $distro/iso/boot/grub/grub.cfg

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

kly ile İso Yapma

Dağıtımlarda, dağıtıma ait paket sistemini kullanarak iso hazırlanabilir. Bu dokümanda paket sistemi(kly) önceki bölümde anlatılmıştı. **kly** paket sistemini kullanarak bir iso nasıl oluşturulur aşağıda verilmiştir. Burada anlatılan iso yapma scripti **Debian** vb. dağıtımlarda da benzer bir yapıdadır. Başka dağıtımlarda iso hazırlayan(deneyimi olan) geliştiriciler iso yapma aşamalırını benzer yapıda olduğunu göreceklerdir.

iso Scripti

```
#!/bin/bash
#set -x
rootfs="/tmp/distro/rootfs"
distro="/tmp/distro"
#rm -rf distro/iso
rm -rf $rootfs
mkdir -p $rootfs
mkdir -p $rootfs/bin $rootfs/etc


##Kurulum scripti
cp -prf files/bin/* $rootfs/bin/
cp -prf files/etc/* $rootfs/etc/

# temel dizinler ve dosyalar oluşturuluyor
cd $rootfs/
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\" >> $rootfs/etc/shell"
bash -c "echo 'tmpfs /tmp tmpfs rw,nodev,nosuid 0 0' >> $rootfs/etc/fstab"
bash -c "echo '127.0.0.1 kly' >> $rootfs/etc/hosts"
bash -c "echo 'kly' > $rootfs/etc/hostname"
bash -c "echo 'nameserver 8.8.8.8' > $rootfs/etc/resolv.conf"

# paket adresi ekleniyor
echo "kendilinuxunuyap/kly-binary-packages">$rootfs/etc/kly/sources.list

### installing kly package in rootfs

echo root:x:0:0:root:/root:/bin/sh > $rootfs/etc/passwd
chmod 755 $rootfs/etc/passwd

### system chroot  bind/mount
for dir in dev dev/pts proc sys; do mount -o bind /$dir $rootfs/$dir; done
## paket listesi güncelleniyor
$rootfs/bin/kly -u $rootfs

## paketler kuruluyor
for paket in glibc readline ncurses \bash openssl acl attr libcap libpcre2 gmp coreutils util-linux \grep \sed mpfr \gawk findutils libgcc libcap-ng \
sqlite \gzip xz-utils zstd \bzip2 \elfutils libselinux \tar \zlib brotli curl shadow \file eudev cpio libsepol \
kmod audit libxcrypt libnsl pam libtirpc e2fsprogs dosfstools  initramfs-tools libxml2 expat libmd libaio lvm2 popt icu iproute2 net-tools  dhcp \
openrc  rsync kbd busybox kernel kernel-headers live-boot live-config parted  nano grub dialog efibootmgr efivar libssh openssh
do
chroot $rootfs /bin/kly -ri $paket;
#$rootfs/bin/kly -ri  $paket $rootfs
done

#### system chroot umount
for dir in dev dev/pts proc sys ; do    while umount -lf -R $rootfs/$dir 2>/dev/null ; do true; done done
exit

chroot $rootfs useradd live -m -s /bin/sh  -d /home/live
chroot $rootfs echo -e "live\nlive\n"|chroot $rootfs passwd live

for grp in users tty wheel cdrom audio dip video plugdev netdev; do
        chroot $rootfs usermod -aG $grp live || true
done

sed -i "/agetty_options/d" $rootfs/etc/conf.d/agetty
echo -e "\nagetty_options=\"-l /usr/bin/login\"" >> $rootfs/etc/conf.d/agetty


### 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

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

#### system chroot umount
for dir in dev dev/pts proc sys ; do    while umount -lf -R $rootfs/$dir 2>/dev/null ; do true; done done

#**********************                                                  iso                                                  ******************************
mkdir -p $distro/iso
mkdir -p $distro/iso/boot
mkdir -p $distro/iso/boot/grub
mkdir -p $distro/iso/live || true

#### Copy kernel and initramfs
cp -pf $rootfs/boot/initrd.img-* $distro/iso/boot/initrd.img
cp -pf $rootfs/boot/vmlinuz-* $distro/iso/boot/vmlinuz
rm -rf $rootfs/boot

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

#### Write grub.cfg
# Timeout for menu
echo -e "set timeout=3\n"> $distro/iso/boot/grub/grub.cfg

# Default boot entry
echo -e "set default=1\n">> $distro/iso/boot/grub/grub.cfg

# Menu Colours
echo -e "set menu_color_normal=white/black\n">> $distro/iso/boot/grub/grub.cfg
echo -e "set menu_color_highlight=white\/blue\n">> $distro/iso/boot/grub/grub.cfg
echo -e "insmod all_video">> $distro/iso/boot/grub/grub.cfg
echo -e "terminal_output console">> $distro/iso/boot/grub/grub.cfg
echo -e "terminal_input console">> $distro/iso/boot/grub/grub.cfg

echo 'menuentry "Canli(live) GNU/Linux 64-bit" --class liveiso  {' >> $distro/iso/boot/grub/grub.cfg
echo '    linux /boot/vmlinuz boot=live init=/sbin/openrc-init net.ifnames=0 biosdevname=0' >> $distro/iso/boot/grub/grub.cfg
echo '    initrd /boot/initrd.img' >> $distro/iso/boot/grub/grub.cfg
echo '}' >> $distro/iso/boot/grub/grub.cfg

echo 'menuentry "Kur GNU/Linux 64-bit" --class liveiso  {' >> $distro/iso/boot/grub/grub.cfg
echo '    linux /boot/vmlinuz boot=live init=/bin/kur quiet' >> $distro/iso/boot/grub/grub.cfg
echo '    initrd /boot/initrd.img' >> $distro/iso/boot/grub/grub.cfg
echo '}' >> $distro/iso/boot/grub/grub.cfg

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

Sistem Kurulumu

Sistem Kurma

Hazırlanmış bir iso ile çeşitli kurulum araçları gelebilir. Bu araçların farklı kurulum yapma yöntemleri olmaktadır. Sık kullanılan yöntemler şunlardır;

1. Tek bölüme sistem kurma
2. iki bölüme sistem kurma(boot+sistem)
3. uefi sistem kurma(boot+sistem)

Burada üç farklı kurulum yöntemi sırayla anlatılacaktır. Bu anlatılan yöntemler birden fazla senaryo isteğine cevap vermektedir. Fakat bu yöntemler için ayrı ayrı seçenek sunarak son kullanıcının kurulum yapılmasını istemek kafa karışıklığına sebep olabilir. Örneğin efi seçeneği olsa fakat sistem legacy olsa veya tam tersi bir durumda kullanıcı bu detayları bilmeyebilir. Ayrıca sda veya nvme disklerde farklı senaryolar gerekecektir. Bizim yapacağımız sistem aşağıdaki tüm senaryolara cevap vermelidir. Muhtemel senaryolar;

- legacy-->sd*
- legacy-->nvme*(desteklenmemektedir)
- uefi-->sd*
- uefi-->nvme* kurulum yapılmak istenebilir.

Bu senaryolara göre sorunsuz çalışacak kurlum scriptimiz **dialog** kullanarak hazırlanmıştır. Bu kurulum scriplerinin **base-file** paketinin **files.rar** paketinde bulunmaktadır. Bu bölümde sadece legacy ve uefi kurulum hangi adımlarla yapıldığını anlatan genel bir yazıdır. Bu yöntemleri kendi ihtiyacınıza göre düzenleyip kullanabilirsiniz.

Tek Bölüm Kurulum

Diskler üzerinde işlem yapabilmek için evdev veya udevd servisi çalışıyor olmalı. Ayrıca aşağıdaki modüllerin yüklü olduğundan emin olun. Anlatım boyunca **/dev/sda** diski üzerinden örnekleme yapılmıştır. Siz kendi diskinize göre düzenleyebilirsiniz. Disk ve isoya erişim için aşağıdaki modüllerin yüklü olduğundan emin olun.

- loop
- squashfs
- ext4 modulleri **modprobe** komutuyla yüklenmeli.

Disk Hazırlanmalı(legacy)

Öncelikle **cfdisk** veya **fdisk** komutları ile diski bölümlendirelim. Ben bu anlatımda **cfdisk** kullanacağım.

1. cfdisk komutuyla disk bölümlendirilmeli.

```
$ cfdisk /dev/sda
```

1. dos seçilmeli
2. type linux system
3. write
4. quit
5. Bu işlem sonucunda sadece sda1 olur
6. mkfs.ext2 ile disk biçimlendirilir.

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

Dosya sistemini kopyalama

Kurulum medyası **/cdrom** dizinine bağlanır. Kurulacak sistemin imajını bir dizine bağlayalım.

```
$ mkdir -p cdrom
$ mkdir -p kaynak
$ mount -t iso9660 -o loop /dev/sr0 /cdrom/
$ mount -t squashfs -o loop /cdrom/live/filesystem.squashfs /kaynak
```

Şimdi de disk bölümümüzü bağlayalım.

```
$ mkdir -p hedef
$ mount /dev/sda1 /hedef
$ mkdir -p /hedef/boot
```

Ardından dosyaları kopyalayalım.

```
# -prfv alt zinlerle beraber dosyanın özniteliklerini koruyarak kopyalar
cp -prfv /kaynak/* /hedef
# diske yazılan bilgiler senkronize edildi.
sync
```

Bootloader kurulumu

grub kurulumu yapmak için grub paketinini kurulu olduğundan emin olun.

```
$ mkdir -p /hedef/dev
$ mkdir -p /hedef/sys
$ mkdir -p /hedef/proc
$ mkdir -p /hedef/run
$ mkdir -p /hedef/tmp
$ mount --bind /dev /hedef/dev
$ mount --bind /sys /hedef/sys
$ mount --bind /proc /hedef/proc
$ mount --bind /run /hedef/run
$ mount --bind /tmp /hedef/tmp

# Bunun yerine aşağıdaki gibi de girilebilir.
for dir in /dev /sys /proc /run /tmp ; do
        mount --bind /$dir /hedef/$dir
done
$ chroot /hedef
```

Grub Kurulumu

```
$ grub-install --boot-directory=/boot  /dev/sda
```

grub.cfg Yapılandırılması

1. /boot bölümünde initrd.img-**kernel-version** dosyamızın olduğundan emin olalım.
2. /boot bölümünde vmlinuz-**kernel-version** kernel dosyamızın olduğundan emin olalım.
3. /boot/grub/grub.cfg konumunda dostamızı oluşturalım(vi, touch veya nano ile).
4. dev/sda1 diskimizim uuid değerimizi bulalım.

```
blkid | grep /dev/sda2
/dev/sda2: UUID="?????" BLOCK_SIZE="xxxxx" TYPE="xxxxx" PARTUUID="xxxxx"
# kernel versiyonu
uname -r
6.1.0-25-amd64
```

Diskimizimin uuid değerine göre /boot/grub/grub.cfg dosyasını aşağıdaki gibi düzenleyip kaydedelim. Burada uuid değerini ve kernel versiyonunu düzenleyelim.

```
linux /boot/vmlinuz-kernel-version      root=UUID= ????? rw quiet
initrd /boot/initrd.img-kernel-version
boot
```

grub.cfg dosyasını elle düzenlemek yerine aşğıdaki komutla otomatik yapılandırılabilir.

```
$ grub-mkconfig -o /boot/grub/grub.cfg
```

İki Bölüm Kurulum

Bu bölümde **Ext4** dosya sistemine grub kullanarak kurulum anlatılacaktır. Anlatım boyunca **/dev/sda** diski kullanılarak anlatılacaktır. Sisteminizdeki diskinize göre düzenleyiniz. Diskler üzerinde işlem yapabilmek için evdev veya udevd servisi çalışıyor olmalı. Disk ve isoya erişim için aşağıdaki modüllerin yüklü olduğundan emin olun.

- loop
- squashfs
- ext4 modulleri **modprobe** komutuyla yüklenmeli.

Disk Hazırlanmalı

Öncelikle **cfdisk** veya **fdisk** komutları ile diski bölümlendirelim. Ben bu anlatımda **cfdisk** kullanacağım.

1. cfdisk komutuyla disk bölümlendirilmeli.

```
cfdisk /dev/sda
```

1. gpt seçilmeli
2. 512 MB type vfat alan(sda1)
3. geri kalanı type linux system(sda2)
4. write
5. quit
6. Bu işlem sonucunda sadece sda1 sda2 olur
7. mkfs.vfat ve mkfs.ext4 ile diskler biçimlendirilir.

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

e2fsprogs Paketi

e2fsprogs paket sistemde mkfs.ext4, e2fsck, tune2fs vb sistem araçlarının yüklenmesini sağlar. Eğer sistemde bu sistem uygulamaları yoksa bu paketin yüklenmesi veya derlenmesi gerekmektedir.

Eğer /boot bölümünü ayırmayacaksanız grub yüklenirken **unknown filesystem** hatası almanız durumunda aşağıdaki yöntemi kullanabilirsiniz.

```
e2fsck -f /dev/sda2
tune2fs -O ^metadata_csum /dev/sda2
```

Dosya Sistemini Kopyalama

Kurulum medyası **/cdrom** dizinine bağlanır. Kurulacak sistemin imajını bir dizine bağlayalım.

```
mkdir -p cdrom
mkdir -p kaynak
mount -t iso9660 -o loop /dev/sr0 /cdrom/
mount -t squashfs -o loop /cdrom/live/filesystem.squashfs /kaynak
```

Şimdi de disk bölümümüzü bağlayalım.

```
mkdir -p hedef
mkdir -p /hedef/boot
mount /dev/sda2 /hedef
mount -t vfat /dev/sda1 /hedef/boot
```

Ardından dosyaları kopyalayalım.

```
# -prfv alt zinlerle beraber dosyanın özniteliklerini koruyarak kopyalar
cp -prfv /kaynak/* /hedef
# diske yazılan bilgiler senkronize edildi.
sync
```

grub Yapılandırılması

grub kurulumu yapmak için grub paketinin kurulu olduğundan emin olun.

```
mkdir -p /hedef/dev
mkdir -p /hedef/sys
mkdir -p /hedef/proc
mkdir -p /hedef/run
mkdir -p /hedef/tmp
mount --bind /dev /hedef/dev
mount --bind /sys /hedef/sys
mount --bind /proc /hedef/proc
mount --bind /run /hedef/run
mount --bind /tmp /hedef/tmp

# Bunun yerine aşağıdaki gibi de girilebilir.
for dir in /dev /sys /proc /run /tmp ; do
mount --bind /$dir /hedef/$dir
done

# chroot /hedef komutuyla hazırladığımız sisteme bağlanıyoruz.
```

Grub Kurulumu

```
# kurulu sistemden bağımsız çalışması için --removable kullanılır.
grub-install --removable --boot-directory=/boot --efi-directory=/boot /dev/sda
```

grub.cfg Yapılandırması

1. /boot bölümünde initrd.img-**kernel-version** dosyamızın olduğundan emin olalım.
2. /boot bölümünde vmlinuz-**kernel-version** kernel dosyamızın olduğundan emin olalım.
3. /boot/grub/grub.cfg konumunda dostamızı oluşturalım(vi, touch veya nano ile).
4. dev/sda2 diskimizim uuid değerimizi bulalım.

```
blkid | grep /dev/sda2
/dev/sda2: UUID="?????" BLOCK_SIZE="xxxxx" TYPE="xxxxx" PARTUUID="xxxxx"
# kernel versiyonu
```

```
uname -r
6.1.0-25-amd64
```

Diskimizimin uuid değerine göre /boot/grub/grub.cfg dosyasını aşağıdaki gibi düzenleyip kaydedelim. Burada uuid değerini ve kernel versiyonunu düzenleyelim.

```
linux /boot/vmlinuz-kernel-version      root=UUID= ????? rw quiet
initrd /boot/initrd.img-kernel-version
boot
```

grub.cfg dosyasını elle düzenlemek yerine aşğıdaki komutla otomatik yapılandırılabilir.

```
grub-mkconfig -o /boot/grub/grub.cfg
```

Uefi Sistem Kurulumu

Bu bölümde **Ext4** dosya sistemine grub kullanarak kurulum anlatılacaktır. Anlatım boyunca **/dev/sda** diski kullanılarak anlatılacaktır. Sisteminizdeki diskinize göre düzenleyiniz. Diskler üzerinde işlem yapabilmek için evdev veya udevd servisi çalışıyor olmalı. Disk ve isoya erişim için aşağıdaki modüllerin yüklü olduğundan emin olun.

- loop
- squashfs
- ext4 modulleri **modprobe** komutuyla yüklenmeli.

Uefi - Legacy Tespiti

/sys/firmware/efi dizini varsa uefi, yoksa legacy sisteme sahipsinizdir. Eğer uefi ise ia32 veya x86_64 olup olmadığını anlamak için **/sys/firmware/efi/fw_platform_size** içeriğine bakın.

```
[[ -d /sys/firmware/efi/ ]] && echo UEFI || echo Legacy
[[ "64" == $(cat/sys/firmware/efi/fw_platform_size) ]] && echo x86_64 || ia32
```

Disk Hazırlanmalı

Uefi kullananlar ayrı bir disk bölümüne ihtiyaç duyarlar. Bu bölümü **fat32** olarak bölümlendirmeliler.

Bu anlatımda kurulum için **/boot** dizinini ayırmayı ve efi bölümü olarak aynı diski kullanmayı tercih edeceğiz. Öncelikle **cfdisk** veya **fdisk** komutları ile diski bölümlendirelim. Ben bu anlatımda **cfdisk** kullanacağım.

1. cfdisk komutuyla disk bölümlendirilmeli.

```
$ cfdisk /dev/sda
```

1. gpt seçilmeli
2. 512 MB type uefi alan(sda1)
3. geri kalanı type linux system(sda2)
4. write
5. quit
6. Bu işlem sonucunda sadece sda1 sda2 olur
7. mkfs.vfat ve mkfs.ext4 ile diskler biçimlendirilir.

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

e2fsprogs Paketi

e2fsprogs paket sistemde mkfs.ext4, e2fsck, tune2fs vb sistem araçlarının yüklenmesini sağlar. Eğer sistemde bu sistem uygulamaları yoksa bu paketin yüklenmesi veya derlenmesi gerekmektedir.

Eğer /boot bölümünü ayırmayacaksanız grub yüklenirken **unknown filesystem** hatası almanız durumunda aşağıdaki yöntemi kullanabilirsiniz.

```
$ e2fsck -f /dev/sda2
$ tune2fs -O ^metadata_csum /dev/sda2
```

Dosya Sistemini Kopyalama

Kurulum medyası **/cdrom** dizinine bağlanır. Kurulacak sistemin imajını bir dizine bağlayalım.

```
$ mkdir -p cdrom
$ mkdir -p kaynak
$ mount -t iso9660 -o loop /dev/sr0 /cdrom/
$ mount -t squashfs -o loop /cdrom/live/filesystem.squashfs /kaynak
```

Şimdi de disk bölümümüzü bağlayalım.

```
mkdir -p hedef || true
mkdir -p /hedef/boot || true
mkdir -p /hedef/boot/efi || true
mount /dev/sda2 /hedef || true
mount /dev/sda1 /hedef/boot/efi
```

Ardından dosyaları kopyalayalım.

```
# -prfv alt zinlerle beraber dosyanın özniteliklerini koruyarak kopyalar
cp -prfv /kaynak/* /hedef
# diske yazılan bilgiler senkronize edildi.
sync
```

grub Yapılandırılması

grub kurulumu yapmak için grub paketinini kurulu olduğundan emin olun.

```
mkdir -p /hedef/dev
mkdir -p /hedef/sys
mkdir -p /hedef/proc
mkdir -p /hedef/run
mkdir -p /hedef/tmp
mount --bind /dev /hedef/dev
mount --bind /sys /hedef/sys
mount --bind /proc /hedef/proc
mount --bind /run /hedef/run
mount --bind /tmp /hedef/tmp

# efi alan bağlanıyor.
# Eğer uefi ise kernel tarafından /sys/firmware/efi dizin ve dosyaları oluşuyor.
# sistem uefi değilse /sys/firmware/efi dosya ve dizini olmayacaktır.
 if [[ -d /sys/firmware/efi ]] ; then
        mount --bind /sys/firmware/efi/efivars /hedef/sys/firmware/efi/efivars
  fi

# Bunun yerine aşağıdaki gibi de girilebilir.
for dir in /dev /sys /proc /run /tmp ; do
        mount --bind /$dir /hedef/$dir
done

# chroot /hedef komutuyla hazırladığımız sisteme bağlanıyoruz.
```

Şimdi de uefi kullandığımız için efivar bağlayalım.

```
mount -t efivarfs efivarfs /sys/firmware/efi/efivarfs
```

Grub Kurulumu

```
# biz /boot ayırdığımız ve efi bölümü olarak kullanacağız.
# uefi kullanmayanlar --efi-directory belirtmemeliler.
# kurulu sistemden bağımsız çalışması için --removable kullanılır.
grub-install --removable --boot-directory=/boot --efi-directory=/boot --target=x86_64-efi /dev/sda
```

grub.cfg Yapılandırması

1. /boot bölümünde initrd.img-**kernel-version** dosyamızın olduğundan emin olalım.
2. /boot bölümünde vmlinuz-**kernel-version** kernel dosyamızın olduğundan emin olalım.
3. /boot/grub/grub.cfg konumunda dostamızı oluşturalım(vi, touch veya nano ile).
4. dev/sda2 diskimizim uuid değerimizi bulalım.

```
blkid | grep /dev/sda2
/dev/sda2: UUID="?????" BLOCK_SIZE="xxxxx" TYPE="xxxxx" PARTUUID="xxxxx"
# kernel versiyonu
uname -r
6.1.0-25-amd64
```

Diskimizimin uuid değerine göre /boot/grub/grub.cfg dosyasını aşağıdaki gibi düzenleyip kaydedelim. Burada uuid değerini ve kernel versiyonunu düzenleyelim.

```
linux /boot/vmlinuz-kernel-version      root=UUID= ????? rw quiet
initrd /boot/initrd.img-kernel-version
boot
```

grub.cfg dosyasını elle düzenlemek yerine aşğıdaki komutla otomatik yapılandırılabilir.

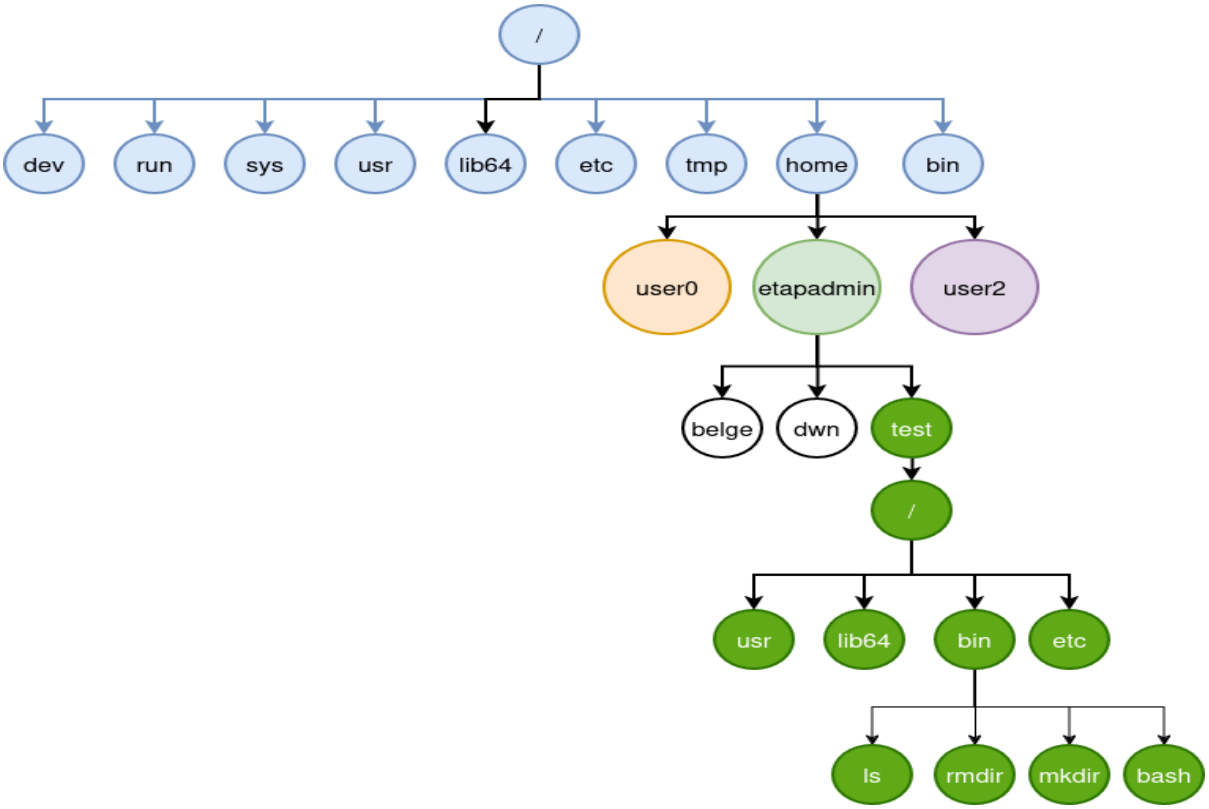
```
grub-mkconfig -o /boot/grub/grub.cfg
```

Yardımcı Konular

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

if [ ${#} != 2 ]
then
    echo "usage $0 PATH_TO_BINARY target_folder"
    exit 1
fi
path_to_binary="$1"
target_folder="$2"

# if we cannot find the the binary we have to abort
if [ ! -f "${path_to_binary}" ]
then
    echo "The file '${path_to_binary}' was not found. Aborting!"
    exit 1
fi

echo "---> copy binary itself" # copy the binary itself
cp --parents -v "${path_to_binary}" "${target_folder}"

echo "---> copy libraries" # copy the library dependencies
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

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

Bu işlemi diğer komutlar içinde sırasıyla yapmamız gerekmektedir.

rmdir Komutu

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

mkdir Komutu

```
# mkdir komutu ve bağımlılığı kopyalandı.
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/

busybox Nedir?

Busybox tek bir dosya halinde bulunan birçok araç seçine sahip olan bir programdır. Bu araçlar initramfs sisteminde ve sistem genelinde sıkça kullanılabilir. Busybox aşağıdaki gibi kullanılır.

```
busybox [komut] (seçenekler)
```

Eğer busyboxu komut adı ile linklersek o komutu doğrudan çalıştırabiliriz. Aşağıda tar uygulamasını busybox dan türettik.

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

Busyboxtaki tüm araçları sisteme sembolik bağ atmak için aşağıdaki gibi bir yol izlenebilir. Bu işlem var olan dosyaları sildiği için tehlikeli olabilir. Sistemin tasarımına uygun olarak yapılmalıdır.

```
# -s parametresi sembolik bağ olarak kurmaya yarar.
busybox --install -s /bin
```

Busybox Derleme

Busybox **static** olarak derlenmediği sürece bir libc kütüphanesine ihtiyaç duyar. initramfs içerisinde kullanılacaksa içerisine libc dahil edilmelidir. Bir dosyanın static olarak derlenip derlenmediğini öğrenmek için aşağıdaki komut kullanılır. Derleme türleri ve detayları için bu dokümandaki derleme konusuna bakınız.

```
# static derlenmişse hata mesajı verir. Derlenmemişse bağımlılıklarını listeler.
ldd /bin/busybox
```

Busybox derlemek için öncelikle **make defconfig** kullanılarak veya önceden oluşturduğumuz yapılandırma dosyasını atarak yapılandırma işlemi yapılır. Ardından eğer static derleme yapacaksak yapılandırma dosyasına müdahale edilir. Son olarak **make** komutu kullanarak derleme işlemi yapılır.

```
make defconfig
sed -i "s|.*CONFIG_STATIC_LIBGCC .*|CONFIG_STATIC_LIBGCC=y|" .config
sed -i "s|.*CONFIG_STATIC .*|CONFIG_STATIC=y|" .config
make
```

Derleme bittiğinde kaynak kodun bulunduğu dizinde busybox dosyamız oluşmuş olur.

Static olarak derlemiş olduğumuz busyboxu kullanarak minimal bir dağıtım hazırlayabiliriz.

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 çalışcak 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 durmunu öğrenmek için **rc-status** komutu kullanılır. Ayrıca sistemdeki servislerin sonraki açılışta hangisinin başlatılacağını öğrenmek için ise 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.

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

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"
command=/usr/bin/ornek-servis
command_args=--parametre
pidfile=/run/ornek-servis.pid
```

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
}
...
```

OpenRc Disk İşlemi

Kullandığımız servis yöneticisi openrc ise **/etc/fstab** komunundaki dosyaya bakarak diske erişim sağlamaktadır. Bundan dolayı **fstab** dosyamızı aşağıdaki gibi yapılandırmalıyız.

Fstab dosyası

Bu dosyayı doldurarak açılışta hangi disklerin bağlanacağını ayarlamalıyız. /etc/fstab dosyasını aşağıdakine uygun olarak doldurun.

```
# <fs>                  <mountpoint>    <type>          <opts>          <dump/pass>
/dev/sda1       /boot   vfat    defaults,rw     0       1
/dev/sda2       /       ext4    defaults,rw     0       1
```

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 doayası 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 belilemek 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:

istemci 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

Live Sistem Oluşturma

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 yazmak 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 oluşturlur. 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 canli
mount -t squashfs -o loop cdrom/live/filesystem.squashfs /canli
```

squashfs Sistemine Geçiş

Yukarıdaki adımlarda squashfs doayamızı /canli adında dizine bağlamış olduk. Bu aşamadan sonra sistemimizin bir kopyası olan squashfs canlıdan erişebilir 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.

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 bir programın veya paylaşılan bir kütüphanenin dosya boyutunu küçültmek için kullanılır. Bu komut, derleme sürecinde oluşturulan fazladan semboller ve hata ayıklama bilgilerini kaldırır. Bu sayede, dosyanın boyutu azalır ve gereksiz veri miktarı azalır.

strip komutunu kullanmak için, terminalde aşağıdaki şekilde bir komut satırı kullanabilirsiniz:

```
strip dosya_adı
```

Burada "dosya_adı" yerine, boyutunu küçültmek istediğiniz dosyanın adını belirtmelisiniz. Örneğin, "strip program" komutunu kullanarak "program" adlı bir dosyanın boyutunu küçültebilirsiniz.

strip komutu, genellikle Linux ve Unix tabanlı işletim sistemlerinde kullanılır. Bu komut, programların dağıtımı veya paylaşımı sırasında dosya boyutunu azaltmak için yaygın olarak kullanılır.

Bu komutun kullanımı, programların performansını artırabilir ve disk alanından tasarruf sağlayabilir. Ancak, hata ayıklama veya sembol bilgilerine ihtiyaç duyduğunuz durumlarda strip komutunu kullanmaktan kaçınmanız önemlidir.

Sonuç olarak, strip komutu, derlenmiş programların veya paylaşılan kütüphanelerin boyutunu küçültmek için kullanılan bir Linux komutudur.

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.
- **Depoyu Oluşturma:** "Create repository" butonuna tıklayarak deponuzu oluşturun.

Artık GitHub üzerinde yeni bir deponuz var! Bu depoyu yerel bilgisayarınıza klonlayarak projelerinizi yönetmeye başlayabilirsiniz. Örneğin, terminalde şu komutu kullanarak deponuzu klonlayabilirsiniz:

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

Bu adımları takip ederek GitHub üzerinde kolayca depo açabilirsiniz.

X11 Kullanıcısının Tespiti

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

```
#!/bin/bash

# Detect the name of the display in use
display=":$(ls /tmp/.X11-unix/* | sed 's#/tmp/.X11-unix/X##' | head -n 1)"

# Detect the user using such display
user=$(who | grep '('$display')' | awk '{print $1}')

# Detect the id of the user
uid=$(id -u $user)

echo $user $uid
```

Hakkımda

Açık kaynaklı sistemleri tanıma ve geliştirme yapma projesi.

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
- Güvenlik en önemli şarttır
- Düşük donanımlarda iyi performan verir
- Bağımlılığı azaltır

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

- https://github.com/kendiliuxunuyap
- kendiliuxunuyap@gmail.com