

distro Dokümanı

version

distro Linux

Ağustos 17, 2024

Contents

Dağıtım	1
Temel Kavramlar	1
Temel Kavramlar	1
Linux Nedir?	1
Açık Kaynak Nedir?	1
Dağıtım Nedir ve Türleri	1
Yazılım Bağımlılığı	1
Ön Hazırlık	2
Dağıtım Ortamın Hazırlanması	2
Minimal Dağıtım Oluşturma	3
Dizinlerin Oluşturulması	3
\$HOME/distro/rootfs/busybox	4
\$HOME/distro/rootfs/init	4
\$HOME/distro/iso/boot/initrd.img	5
\$HOME/distro/iso/boot/vmlinuz	5
\$HOME/distro/iso/boot/grub/grub.cfg	5
\$HOME/distro/distro.iso	5
Dağıtımın Test Edilmesi	5
Derleme Araçları	7
Derleme araçları	7
make	7
cmake	8
meson	8
python	9
Paket Derleme	9
Temel Paketler	9
base-file	12
base-file Komutları	12
Şablon Script Yapısı	13
Yapıyı Oluşturan Script	14
Paket Derleme Yöntemi	15
glibc	16
Derleme	16
Yükleme	16
glibc Script Dosyası	16
Test Etme	18
Program Derleme	18
Program Yükleme	18
Programı Test Etme	19
Hata Çözümü	19
libreadline	21

Derleme	21
Program Yazma	22
Program Derleme	22
Program Test Etme	22
ncurses	23
Derleme	23
bash	25
Derleme	25
openssl	27
Derleme	27
acl	29
Derleme	29
attr	30
Derleme	30
libcap	31
Derleme	32
libcap-ng	34
Derleme	34
gmp	35
Derleme	35
grep	36
Derleme	36
sed	37
Derleme	37
mpfr	38
Derleme	38
gawk	39
Derleme	39
findutils	40
Derleme	40
libpcre2	41
Derleme	41
coreutils	42
Derleme	42
util-linux	44
Derleme	45
gzip	47
Derleme	47
xz-utils	48
Derleme	48
zstd	49
Derleme	49
bzip2	50

Derleme	51
elfutils	53
Derleme	53
libselinux	55
Derleme	55
libsepol	57
Derleme	57
tar	58
Derleme	58
zlib	59
Derleme	59
brotli	60
Derleme	60
curl	61
Derleme	62
libxml2	64
Derleme	64
eudev	66
Derleme	67
libnsl	69
Derleme	69
pam	70
Derleme	70
expat	72
Derleme	72
libmd	74
Derleme	74
audit	75
Derleme	76
libgcc	78
Derleme	79
libxcrypt	81
Derleme	81
sqlite	82
Derleme	82
libtirpc	84
Derleme	84
file	85
Derleme	85
e2fsprogs	87
Derleme	87
dostools	88
Derleme	88

initramfs-tools	89
Derleme	90
/etc/initramfs-tools/modules	91
initramfs-tools Ayarları	91
initramfs-tools Güncelleme	92
initrd açılma Süreci	92
initrd script İçeriği	92
cpio	93
Derleme	93
libio	94
Derleme	94
lvm2	95
Derleme	96
popt	98
Derleme	98
icu	99
Derleme	99
iproute2	100
Derleme	100
net-tools	102
Derleme	102
dhcp	104
Derleme	104
shadow	106
Derleme	107
openrc	109
Derleme	110
Çalıştırılması	111
Basit kullanım	111
rsync	113
Derleme	113
kbd	114
Derleme	114
busybox	116
Derleme	117
kernel	119
Derleme	120
kernel-headers	123
Derleme	124
live-boot	127
Derleme	127
live-config	128
Derleme	128

parted	129
Derleme	129
kmod	130
Derleme	130
kmod Komutları	131
Test Edilmesi	131
nano	132
Derleme	132
grub	133
Derleme	134
efibootmgr	136
Derleme	136
efivar	137
Derleme	137
dialog	139
Derleme	139
libssh	140
Derleme	140
openssh	142
Derleme	142
Paket Sistemi	144
Paket Sitemi	144
Binary Paket Sistemi	144
Source Paket Sistemi	144
Paket sisteminin temel yapısı	144
bps Paket Sistemi	144
Paket Oluşturma	145
bpsbuild Dosyası	145
bpspaketle Dosyası	145
bpsbuild Dosyamızın Son Hali	146
bpspaketle Dosyamızın Son Hali	147
Paket Yapma	148
Depo indexleme	149
bps Paket Liste İndexi Güncelleme	149
index.lst Dosyasını Oluşturma	150
index.lst Dosyasını Güncelleme	150
Paket Kurma	151
bps Paket Kurma Scripti Tasarlama	151
bpskur Scripti	152
bpskur Scriptini Kullanma	152
Paket Kaldırma	153
bps Paket Kaldırma Scripti Tasarlama	153
bpskaldir scripti	153

bpskaldır Kullanma	154
iso Hazırlama	155
İso Hazırlama	155
filesystem.squashfs Hazırlama	155
İso Dosyasının Oluşturulması	155
Sistem Kurulumu	156
Sistem Kurma	156
Tek Bölüm Kurulum	157
Disk Hazırlanmalı(legacy)	157
Dosya sistemini kopyalama	157
Bootloader kurulumu	158
Grub Kuralım	158
Grub yapılandırması	158
OpenRc Disk İşlemi	159
Fstab dosyası	159
İki Bölüm Kurulum	160
Disk Hazırlanmalı	160
e2fsprogs Paketi	160
Dosya sistemini kopyalama	160
Bootloader kurulumu	161
Grub Kuralım	161
Grub yapılandırması	161
OpenRc Disk İşlemi	162
Fstab dosyası	162
Tek Bölüm Kurulum	163
Disk Hazırlanmalı(legacy)	163
Dosya sistemini kopyalama	163
Bootloader kurulumu	164
Grub Kuralım	164
Grub yapılandırması	164
OpenRc Disk İşlemi	165
Fstab dosyası	165
Uefi Sistem Kurulumu	166
Uefi - Legacy tespiti	166
Disk Hazırlanmalı	166
e2fsprogs Paketi	166
Dosya sistemini kopyalama	167
Bootloader kurulumu	167
Grub Kuralım	168
Grub yapılandırması	168
OpenRc Disk İşlemi	168
Fstab dosyası	168
Uefi Sistem Kurulumu	169

Uefi - Legacy tespiti	169
Disk Hazırlanmalı	169
e2fsprogs Paketi	169
Dosya sistemini kopyalama	170
Bootloader kurulumu	170
Grub Kuralım	171
Grub yapılandırması	171
OpenRc Disk İşlemi	171
Fstab dosyası	171
Yardımcı Konular	172
Chroot Nedir?	172
Bağımlılık Scripti	173
Basit Sistem Oluşturma	173
Sistem Dizinini Oluşturulması	174
Is Komutu	174
rmdir Komutu	174
mkdir Komutu	175
bash Komutu	175
chroot Sistemde Çalışma	175
Qemu Kullanımı	176
Qemu Nedir?	176
Sisteme Kurulum	176
Sistem Hızlandırılması	176
Boot Menu Açma	176
Uefi kurulum için:	176
qemu Host Erişimi:	177
vmlinuz ve initrd	177
qemu Terminal Yönlendirmesi	177
Diskteki Sistemin Açılışını Terminale Yönlendirme	177
Live Sistem Oluşturma	178
SquashFS Nedir?	178
SquashFS Oluşturma	178
Cdrom Erişimi	178
Cdrom Bağlama	178
squashfs Dosyasını Bulma	178
squashfs Bağlama	178
squashfs Sistemine Geçiş	178
kmod Nedir?	180
Modul Yazma	180
hello.c dosyamız	180
Makefile dosyamız	181
modülün derlenmesi ve eklenip kaldırılması	181
Not:	181

sfdisk Nedir	182
Disk Bölümlerini Görüntüleme:	182
Disk Bölümleri Oluşturma:	182
Disk Bölümlerini Silme:	182
Disk Etiketi Belirleme	182
Bazı Örnekler	182
Örnek1:	182
Örnek2:	183
Örnek3:	183
Örnek4:	183
İmza Doğrulama	184
İmza Oluşturma	184
Belge İmzalama	184
İmzalı Belge Doğrulama	184
bash ile Doğrulama	184
c++ ile Doğrulama	186
c++ libpgpme ile Doğrulama	188
Kernel Modul Derleme	191
Kaynak Dosya İndirme	191
Kernel Ayarları	191
Kernel Derleme	191
Modul Derleme	191
Modül Yükleme	192
Strip Yapma	192
Kernel Yükleme	192
Header Yükleme	192
libc Yükleme	192
busybox Nedir?	193
Busybox Derleme	193
initrd Tasarımı(Debian)	194
Sistem İçin Gerekli Olan Dosyalar Ve Açılış Süreci	194
initrd Nedir? Nasıl Hazırlanır?	194
Dizin Yapısının oluşturulması	195
S1- distro/initrd/bin/busybox	195
S2-S8 distro/initrd/bin/kmod	195
S9- distro/initrd/lib/modules/\$(uname -r)/moduller	196
S9- distro/initrd/bin/systemd-udevd	196
S10- distro/initrd/bin/udevadm	196
S12- distro/etc/udev/rules.d--S13- distro/lib/udev/rules.d	197
S14- distro/initrd/bin/init	197
S15- distro/iso/initrd.img - S16- distro/iso/vmlinuz	198
S17- distro/iso/grub/grub.cfg	198
İso Oluşturma	198

Bağımlılıkların Tespiti	198
initrd(bps)	200
Temel Dosyalar	200
linux Açılış Süreci	200
initrd Dosya İçeriği	201
initrd Scripti	202
S1- \$boot/bin/busybox	202
S2-S8 \$boot/bin/kmod	203
S9- \$boot/lib/modules/\$(uname -r)/moduller	203
S10-S13- \$boot/bin/udevadm	203
S14- distro/initrd/bin/init	204
init Dosyası	204
S15- distro/iso/initrd.img	205
S16- distro/iso/vmlinuz	205
S17- distro/iso/grub/grub.cfg	205
strip	206
ld(linker)	207
Linker Türleri	207
cmake	208
sudoers	209
polkit	210
Tüm Uygulamalarda İzin Verme	210
Bir Gruba İzin Verme	210
Bir Grub-User-Uygulamaya İzin Verme	211
user-dirs	212
Sistem Dili Değiştirme	213
1. Yöntem	213
2. Yöntem	213
profile	213
3.Yöntem	214
Kullanıcı Sistem Ayarları	214
Profile Dosyası	214
** profile**	214
adduser ve useradd Kullanımı	214
internet	215
Terminal Yönlendirmesi	215
OpenRC	217
Kurulum	217
Çalıştırılması	217
Basit kullanım	217
Servis dosyası	218

Dağıtım

Temel Kavramlar

Temel Kavramlar

Linux Nedir?

Linux; Linux çekirdeğine dayalı, açık kaynak kodlu, Unix benzeri bir işletim sistemi ailesidir. GNU Genel Kamu Lisansı versiyon 2 ile sunulan ve Linux Vakfı çatısı altında geliştirilen bir özgür yazılım projesidir. Linux, "Linus Torvalds" tarafından geliştirilmeye başlanmıştır. Günümüzde bilgisayarlarda, akıllı cihazların ve internet altyapısında kullanılan cihazların işletim sistemlerinde yaygın olarak kullanılmaktadır.

Açık Kaynak Nedir?

Açık kaynak terimi, yazılım geliştirme sürecinde kaynak kodunun genel olarak erişilebilir ve değiştirilebilir olduğu bir modeli ifade eder. Bu yaklaşım, yazılımın geliştirilmesine katkıda bulunanların kodu incelemesine, değiştirmesine ve geliştirmesine olanak tanır. Açık kaynaklı yazılımlar genellikle topluluklar tarafından desteklenir ve geliştirilir, bu da geniş bir bilgi ve deneyim havuzundan faydalanmayı sağlar. Bu model, yazılımın sürekli olarak gelişmesine ve iyileştirilmesine olanak tanırken, kullanıcılar için de esneklik ve özgürlük sağlar.

Dağıtım Nedir ve Türleri

GNU, uygulamalar, kütüphaneler, geliştirici araçları, hatta oyunların bir arada olduğu Unix benzeri bir işletim sistemidir. Linux çekirdeği kullanılarak oluşturulan işletim sistemlerini tanımlamak için yaygın olarak Linux ismi kullanılmaktadır. Mesela, Linux çekirdeği ve GNU araçları bir araya getirilerek oluşturulan bir işletim sistemi "GNU/Linux dağıtımı" olarak adlandırılır, ancak konuşma dilinde kısaca (yanlış da olsa) Linux olarak ifade edilmektedir.

1. Clonezilla İmaj(tahta imajları)
2. Dağıtım Editleme(etahta)
3. Paket sistemi yazma (ahenk,debian)

Yazılım Bağımlılığı

İşletim sistemi tekeli, belirli bir işletim sisteminin pazar payının diğer işletim sistemlerine göre belirgin şekilde yüksek olması durumunu ifade eder. Bu durumda, belirli bir işletim sistemi diğer alternatiflerine kıyasla daha yaygın olarak kullanılır ve genellikle bu işletim sisteminin sahibi veya geliştiricisi, pazarda belirleyici bir konuma sahip olabilir.

Linux gibi açık kaynaklı işletim sistemleri, işletim sistemi tekeline alternatif oluşturabilir ve çeşitliliği artırabilir. Bu sayede kullanıcılar farklı seçeneklere sahip olabilir ve rekabet teşvik edilebilir.

İşletim sistemi tekeli kavramı, pazar analizlerinde, rekabet politikalarında ve tüketici hakları açısından önemli bir konudur. Bu bağlamda, işletim sistemi çeşitliliği ve rekabetin desteklenmesi, kullanıcıların daha iyi hizmet ve ürün seçeneklerine erişimini sağlayabilir.

Ön Hazırlık

Dağıtım Ortamın Hazırlanması

Dağıtım hazırlarken sistemin derlenmesi ve gerekli ayarlamaların yapılabilmesi için bir linux dağıtımı gerekmektedir. Tecrübeli olduğunuz bir dağıtımı seçmenizi tavsiye ederim. Fakat seçilecek dağıtım Gentoo olması daha hızlı ve sorunsuz sürece devam etmenizi sağlayacaktır. Bu dağıtımı hazırlarken 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 için 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 kullanıcının ev dizinini ifade eder. Örneğin sisteme giriş yapan kullanıcı **ogrenci** adında bir kullanıcı olsun. **\$HOME** ifadesi **/home/ogrenci** konumunu ifade eder.

Minimal Dağıtım Oluşturma

Bu minimal dağıtım 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/vmlinuz** 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 oluşturmamız gerekir.

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 belirtilen 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 gerekmekeydi.

\$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 kullanılabilir. Bulduğumuz 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.

Ön Hazırlık

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.

```
[ 5.034768] evm: security.apparmor
[ 5.034768] evm: security.ima
[ 5.034768] evm: security.capability
[ 5.034768] evm: HMAC attrs: 0x1
[ 6.042439] Unstable clock detected, switching default tracing clock to "global"
[ 6.042439] If you want to keep using the local clock, then add:
[ 6.042439] "trace_clock=local"
[ 6.042439] on the kernel command line
[ 6.102953] Freeing unused decrypted memory: 2036K
[ 6.586777] Freeing unused kernel image (initmem) memory: 2792K
[ 6.593434] Write protecting the kernel read-only data: 26624k
[ 6.611100] Freeing unused kernel image (text/rodata gap) memory: 2040K
[ 6.611100] Freeing unused kernel image (rodata/data gap) memory: 1184K
[ 7.522800] x86/mm: Checked W*X mappings: passed, no W*X pages found.
[ 7.522800] Run /init as init process

BusyBox v1.35.0 (Debian 1:1.35.0-4+b3) built-in shell (ash)
Enter 'help' for a list of built-in commands.

ash: can't access tty: job control turned off
/ # ls/b'n
> ls/bin
>
```

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 gömümlü (dahil) bir şekilde gelir. Avantajı hiçbir kütüphaneye ihtiyaç duymaz. Deventajı ise boyutları yüksek olur. İstisnalar olsada genel olarak static tercih edilmemektedir. Static olduğunda iso boyutlarımız olması gerekenden 5-10 kat fazla olabilmektedir.

Derleme Araçları

Derleme araçları

Linux dağıtımlarında kaynak kodlar bir derleme aracı yardımı ile derlenir. Bu araçlar sayesinde kaynak kod tek tek komut yazmaya gerek kalmadan derlenebilir. Ayrıca hataları yakalaması da daha kolay olur.

Bu bölümde başlıca kullanılan derleme araçlarından söz edilecektir. Bu araçlar şunlardır;

- make
- cmake
- meson
- python

make

Bu derleme yöntemi aşağıda gösterilen örnekteki gibi kullanılır.

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

Bu örnekte ilk satırda kullanılan configure komutu **autotools** paketinden gelmektedir. Eğer configure dosyası yerine **configure.ac** varsa önce **autoreconf -fvi** çalıştırılmalıdır. Bu komut configure dosyamızı üretir. **configure** komutunun devamında **--prefix** dışında başka parametreleride olabilir. Bu parametreleri "Read.me" dosyası içerisinde bulunabilir veya **configure --help** komutu kaynak kodların olduğu konumda kullanılarak görülebilir.

Örneğin aşağıdaki gibi bir C dosyamız olsun:

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

configure.ac dosyamızın içeriği aşağıdaki gibi olsun:

```
# initialize the process
AC_INIT([hello], [0.01])
# make config headers
AC_CONFIG_HEADERS([config.h])
#Auxiliary files go here
AC_CONFIG_AUX_DIR([build-aux])
# init automake
AM_INIT_AUTOMAKE([1.11])
#configure and create "Makefile"
AC_CONFIG_FILES([Makefile])
#find and probe C compiler
AC_PROG_CC
#End
AC_OUTPUT
```

makefile.am dosyamızın içeriği aşağıdaki gibi olsun:

Derleme Araçları

```
#list of programs to be installed in bin directory
bin_PROGRAMS = hello
#sources for targets
hello_SOURCES = hello.c
```

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 araçtır. 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.

Paket Derleme

Bu kaynak kod aşağıdaki gibi derlenir:

```
DESTDIR=/ #paketin yükleneceği konum
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:

```
DESTDIR=/ #yükleme konumu
python3 -m build
python3 -m installer --destdir="$DESTDIR" dist/*.whl
```

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

Paket Derleme

Temel Paketler

Bir önceki bölümde minimal bir sistemi **busybox** yardımıyla tasarladık. Minimal sistem tasarımımda 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 bağlanabilir(udhcpc).
- Dosya indirebilir(wget, curl vb.)
- Log(raporlama) tutabilir.
- Modülleri yönetebilir(kmod, modprobe,insmod vb.)

Paket Derleme

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

Paket Derleme

23- elfutils	48- popt	73-
24- libselinux	49- icu	74-

Listede bulunan **bash** paketinin sorunsuz çalışabilmesi için **readline** ve **ncurses** kütüphaneleri gerekli. **readline** ve **ncurses** kütüphanelerinin çalışabilmesi içinde **glibc** kütüphanesi gerekli. Listede bulunan tüm paketlerin bağımlılıkları eksiksizdir. Listede bulunan paketler sırasıyla nasıl derleneceği ayrı başlıklar altında anlatılacaktır.

Paket Derleme

base-file

Bir sistem tasarımı için temel ayarlamalar, dosya ve izin yapıları olması gerekmektedir. Bu yapıyı oluşturduktan sonra sistemi bu yapının üzerine inşa edeceğiz. Sistemin ilk ve temel ayarları olduğundan dolayı **base-file** ismini kullandık. Aslında linux sisteminde temel paket **glibc** paketidir. **glibc** paketinin derlenip yüklenmesinden önce temel yapının oluşturulması gerektiği için **base-file** paketi oluşturduk.

base-file Komutları

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

mkdir -p bin dev etc home lib64 proc root run sbin sys usr var etc/bps tmp tmp/bsp/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 \"\n/bin/sh\n/bin/bash\n/bin/rbash\n/bin/dash\" >> $HOME/distro/build/etc/shell"
bash -c "echo 'tmpfs /tmp tmpfs rw,nodev,nosuid 0 0' >> $HOME/distro/build/etc/fstab"
bash -c "echo '127.0.0.1 basitdagitim' >> $HOME/distro/build/etc/hosts"
bash -c "echo 'basitdagitim' > $HOME/distro/build/etc/hostname"
bash -c "echo 'nameserver 8.8.8.8' > $HOME/distro/build/etc/resolv.conf"

echo root:x:0:0:root:/root:/bin/sh > $HOME/distro/build/etc/passwd
chmod 755 $HOME/distro/build/etc/passwd

cp -prfv $HOME/distro/build/* $HOME/distro/rootfs/
```

Bu komutlar yöntem olarak doğru olsada daha fonksiyonel hale getirmek için aşağıda verilen script şablon yapısını kullanacağız.

Paket Derleme

Şablon Script Yapısı

```
#!/usr/bin/env bash
version="1.0"
name="base-file"
depends=""
description=""
source=""
groups="sys.base"
BUILDDIR="$HOME/distro/build" #Derleme yapılan dizin
DESTDIR="$HOME/distro/rootfs" #paketin yükleneceği sistem konumu
PACKAGEDIR=$(pwd)
SOURCEDIR="$BUILDDIR/${name}-${version}"
initsetup(){
    # Paketin kaynak dosyalarının indirilmesi
    mkdir -p $BUILDDIR #derleme dizini yoksa oluşturuluyor
    mkdir -p $DESTDIR #paketin yükleneceği sistem konumu yok oluşturuluyor
    rm -rf $BUILDDIR/* #içeriği temizleniyor
    cd $BUILDDIR #dizinine geçiyoruz
    wget ${source}
    downloadfile=$(ls|head -1)
    filetype=$(file -b --extension $downloadfile|cut -d'/' -f1)
    if [ "${filetype}" == "???" ]; then unzip ${downloadfile}; else tar -xvf ${downloadfile};fi
    director=$(find ./ -maxdepth 0 -type d)
    mv $director ${name}-${version};
}
setup(){
    #Derleme öncesi kaynak dosyaların sisteme göre ayarlanması
}
build(){
    #Paketin derlenmesi
}
package(){
    # Derlenen dosyaları yükleme öncesi ayar ve yükleme işleminin yapılması
}

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

Şablon dosyasındaki her bir fonksiyonu aslında **base-file** için paylaşılan komutları adım adım yaptığımız işlemleri kapsamaktadır. Biz bu işlem adımlarını şablon dosyamızın ilgili fonksiyonlarına aşama aşama yaptığımız işlemleri ayıracağız. **base-file** script dosyasına benzer yapıda diğer paketler içinde script dosyası oluşturulacaktır. Bu sayede her aşamayı tek tek yazma gibi iş yükü olmayacak ve paket derlenirken hangi fonksiyonda (initsetup, setup vb.) sorun yaşanırsa o fonksiyon üzerinden hata ayıklama yapılacaktır. Bu şekilde bir script dosyasına ileri aşamalarda daha yeni özellikler katma ve kontrol etmeye imkan sağlayacaktır. **base-file** scriptide dahil sonraki aşamalarda yapacağınız çalıştıracağınız script dosyaları bir dizin içinde sırasıyla (1-base-file vb) saklamanızı tavsiye ederim. Daha sonra bu işlemleri tekrarlamanız durumunda hangi sırayla paketleri derleyeceğinizi anlamanız ve hızlıca paketleri derlemenizi kolaylaştıracaktır.

Paket Derleme

Yapıyı Oluşturan Script

```
#!/usr/bin/env bash
version="1.0"
name="base-file"
depends=""
description="sistemin temel yapısı"
source=""
groups="sys.base"
ROOTBUILDDIR="$HOME/distro/build"
BUILDDIR="$HOME/distro/build/build-{$name}-{$version}" #Derleme yapılan dizin
DESTDIR="$HOME/distro/rootfs" #Paketin yükleneceği sistem konumu
PACKAGEDIR=$(pwd)
SOURCEDIR="$HOME/distro/build/{$name}-{$version}"
initsetup(){
# Paketin kaynak dosyalarının indirilmesi
mkdir -p $ROOTBUILDDIR #derleme dizini yoksa oluşturuluyor
rm -rf $ROOTBUILDDIR/* #içeriği temizleniyor
mkdir -p $ROOTBUILDDIR #paketin yükleneceği sistem konumu yok oluşturuluyor
cd $BUILDDIR #dizinine geçiyoruz
}
setup(){
    cp -prfv $PACKAGEDIR/files/* $BUILDDIR/
    cd $BUILDDIR #dizinine geçiyoruz
    echo ""
}
build(){
    echo ""
}
package(){
    mkdir -p bin dev etc home lib64 proc root run sbin sys usr var etc/tps tmp tmp/tps/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 \"\n/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 basitdagitim' >> $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/
}
initsetup      # initsetup fonksiyonunu çalıştırır ve kaynak dosyayı indirir
setup          # setup fonksiyonu çalışır ve derleme öncesi kaynak dosyaların ayarlanması sağlanır.
build          # build fonksiyonu çalışır ve kaynak dosyaları derlenir.
package        # package fonksiyonu çalışır, yükleme öncesi ayarlamalar yapılır ve yüklenir.
```

Yukarıdaki kodların sorunsuz çalışabilmesi için ek dosyalara 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 açınız.

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
./build
```

Paket Derleme

Paket Derleme Yöntemi

base-file paketleri ilk paketler olmasından dolayı detaylıca anlatıldı. Bu paketten sonraki paketlerde **şablon script** dosyası yapıda 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. Derleme scripti için **build** dosyası oluşturup içine yapıştırın ve kaydedin. **build** dosyasının bulunduğu dizininde terminali açarak aşağıdaki gibi çalıştırınız.

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

Paket Derleme

glibc

glibc dağıtımda sistemdeki bütün uygulamaların çalışmasını sağlayan en temel C kütüphanesidir. GNU C Library(glibc)'den farklı diğer C standart kütüphaneler şunlardır: Bionic libc, dietlibc, EGLIBC, klibc, musl, Newlib ve uClibc. **glibc** yerine alternatif olarak çeşitli avantajlarından dolayı kullanılabilir. **glibc** en çok tercih edilen ve uygulama (özgür olmayan) uyumluluğu bulunduğu için bu dokümanda glibc üzerinden anlatım yapılacaktır.

glibc (GNU C Kütüphane) Linux sistemlerinde kullanılan bir C kütüphanesidir. Bu kütüphane, C programlama dilinin temel işlevlerini sağlar ve Linux çekirdeğiyle etkileşimde bulunur. **glibc** listemizdeki tüm paketlerin çalışması için temel kütüphanedir. Bundan dolayı ilk olarak derleyeceğiz pakettir.

Derleme

```
mkdir -p /$HOME/distro/build #derleme dizini yoksa oluşturuluyor
rm -rf /$HOME/distro/build/* #içeriği temizleniyor
cd /$HOME/distro/build #dizinine geçiyoruz
wget https://ftp.gnu.org/gnu/libc/glibc-2.38.tar.gz # glibc kaynak kodunu indiriyoruz.
tar -xvf glibc-2.38.tar.gz # glibc kaynak kodunu açıyoruz.
cd glibc-2.38
configure --prefix=/ --disable-werror # Derleme ayarları yapılıyor
make # glibc derleyelim.
```

Yükleme

```
make install DESTDIR=$HOME/distro/rootfs # Ev Dizinindeki /distro/rootfs dizinine glibc yükleyelim.
```

glibc Script Dosyası

Debian ortamında bu paketin derlenmesi için; **sudo apt install make autotools gawk diffutils gcc gettext grep perl sed texinfo** 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++",
ROOTBUILDDIR="$HOME/distro/build"
BUILDDIR="$HOME/distro/build/build-${name}-${version}" #Derleme yapılan dizin
DESTDIR="$HOME/distro/rootfs" #Paketin yükleneceği sistem konumu
PACKAGEDIR=$(pwd)
SOURCEDIR="$HOME/distro/build/${name}-${version}"
initsetup(){
    mkdir -p $ROOTBUILDDIR #derleme dizini yoksa oluşturuluyor
    rm -rf $ROOTBUILDDIR/* #içeriği temizleniyor
    cd $ROOTBUILDDIR #dizinine geçiyoruz
    wget ${source}
    downloadfile=$(ls|head -1)
    filetype=$(file -b --extension $downloadfile|cut -d'/' -f1)
    if [ "${filetype}" == "???" ]; then unzip ${downloadfile}; else tar -xvf ${downloadfile};fi

    director=$(find ./ -maxdepth 0 -type d)
    if [ "${director}" != "${name}-${version}" ]; then mv $director ${name}-${version};fi
    cd $BUILDDIR
}
```

Paket Derleme

```
setup()
{
    cp -prvf $PACKAGEDIR/files $BUILDDIR/

    opts=(
        --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
    )

    echo "slibdir=/lib64" >> configparms
    echo "rtlddir=/lib64" >> configparms
    $BUILDDIR/${name}-${version}/configure ${opts[@]} \
    --host=x86_64-pc-linux-gnu \
    --libdir=/lib64 \
    --libexecdir=/lib64/glibc
}

build()
{
    make -j5 #-C $DESTDIR all
}

package()
{
    #cd $SOURCEDIR
    # create symlink lib64 (gentoo compability)
    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/
    # remove ld.so.cache file (this file must generated by ldconfig command from ymp)
    rm -f ${DESTDIR}/etc/ld.so.cache
    # install sysconf trigger

    install $BUILDDIR/files/glibc.sysconf ${DESTDIR}/etc/sysconf.d/glibc
    # install extra tools
    install $BUILDDIR/files/locale-gen ${DESTDIR}/bin/locale-gen
    install $BUILDDIR/files/revdep-rebuild ${DESTDIR}/bin/revdep-rebuild
    # replace buggy turkish format with better one
    install $BUILDDIR/files/tr_TR ${DESTDIR}/usr/share/i18n/locales/tr_TR
    # remove unused languages
    for l in ku hy ; do
        rm -rf ${DESTDIR}/usr/lib/locale/${l}_*
        rm -rf ${DESTDIR}/usr/share/locale/${l}_*
```

Paket Derleme

```
rm -rf ${DESTDIR}/usr/share/i18n/locales/${i}_*
done
# fix ldd shebang
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 -r file; do
    ln -s $file $(basename "$file")
done< <(find "../" -maxdepth 1 -type f -iname "*" -print0)

}

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

Yukarıdaki kodların sorunsuz çalışabilmesi için ek dosyalara ihtiyaç vardır. Bu ek dosyaları indirmek için [tıklayınız](#).

tar dosyasını indirdikten sonra **glibc** adında bir dizin oluşturun ve tar dosyasını oluşturulan dizin içinde açınız.

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 **glibc** dizinin içinde terminal açarak çalıştırınız.

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

Test Etme

glibc kütüphanemizi **\$HOME/distro/rootfs** konumuna yükledik. 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

Aşağıdaki komutlarla merhaba.c dosyası derlenir.

```
cd $HOME
gcc -o merhaba merhaba.c
```

Program Yükleme

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

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

Paket Derleme

Programı Test Etme

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

Aşağıdaki gibi çalıştırdığımızda bir hata alacağız.

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

Hata Çözümü

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

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

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

Merhaba Dünya mesajını gördüğümüzde glibc kütüphanemizin ve merhaba çalışabilir dosyamızın çalıştığını anlıyoruz. Bu aşamadan sonra **Temel Paketler** listemizde bulunan paketleri kodlarından derleyerek **\$HOME/distro/rootfs/** dağıtım dizinimize yüklemeliyiz. Derlemede **glibc** kütüphanesinin derlemesine benzer bir yol izlenecektir. **glibc** temel kütüphane olması ve ilk derlediğimiz paket olduğu için detaylıca anlatılmıştır.

glibc kütüphanemizi derlerken yukarıda yapılan işlem adımlarını ve hata çözümlemesini bir script dosyasında yapabiliriz. Bu dokümanda altta paylaşılan script dosyası yöntemi tercih edildi. Aslında yukarıdaki işlem adımlarının aynısını bir dosya içerisine eklemiş olduk. Tek tek çalıştırmak yerine bir script dosya içine eklemeyerek tek bir işlem adımıyla tüm aşamalar çalıştırılabilir.

```
# tanımlamalar
version="2.38"
name="glibc"

# derleme yerinin hazırlanması
mkdir -p $HOME/distro/build #derleme dizini yoksa oluşturuluyor
rm -rf $HOME/distro/build/* #içeriği temizleniyor
cd $HOME/distro/build #dizinine geçiyoruz
wget https://ftp.gnu.org/gnu/libc/${name}-${version}.tar.gz
tar -xvf ${name}-${version}.tar.gz
cd ${name}-${version} # Kaynak kodun içine giriliyor

# derleme öncesi paketin ayarlanması
./configure --prefix=/ --disable-werror

# derleme
make

# derlenen paketin yüklenmesi ve ayarlamaların yapılması
make install DESTDIR=$HOME/distro/rootfs
cd $HOME/distro/rootfs/
ln -s lib lib64
```

Paket Derleme

Diğer paketlerimizde de **glibc** için paylaşılan script dosyası gibi dosyalar hazırlayıp derlenecektir. Yukarıda paylaşılan **script** dosya tekrar düzenlenerek aşağıda son haline getirilecektir. Aşağıda paylaşılan **script** dosya üstteki script dosyadan bir farkı yok. Sadece fonksiyonel hale getirilerek daha anlaşılır ve kontrol edilebilir hale getiriyoruz. Son halinin şablon script dosyası ve ona uygun **glibc** scriptinini hazırlanmış hali aşağıda verilmiştir.

Paket Derleme

libreadline

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.

Derleme

```
#!/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"
ROOTBUILDDIR="$HOME/distro/build"
BUILDDIR="$HOME/distro/build/build-${name}-${version}" #Derleme yapılan dizin
DESTDIR="$HOME/distro/rootfs" #Paketin yükleneceği sistem konumu
PACKAGEDIR=$(pwd)
SOURCEDIR="$HOME/distro/build/${name}-${version}"
initsetup(){
    mkdir -p $ROOTBUILDDIR #derleme dizini yoksa oluşturuluyor
    rm -rf $ROOTBUILDDIR/* #içeriği temizleniyor
    cd $ROOTBUILDDIR #dizinine geçiyoruz
    wget ${source}
    downloadfile=$(ls|head -1)
    filetype=$(file -b --extension $downloadfile|cut -d'/' -f1)
    if [ "${filetype}" == "???" ]; then unzip ${downloadfile}; else tar -xvf ${downloadfile};fi
    director=$(find ./ -maxdepth 0 -type d)
    if [ "${director}" != "${name}-${version}" ]; then mv $director ${name}-${version};fi
    cd $BUILDDIR
}

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

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

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

    install -Dm644 ../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.
```

Yukarıdaki kodların sorunsuz çalışabilmesi için ek dosyalara ihtiyaç vardır. Bu ek dosyaları indirmek için [tıklayınız](#).

tar dosyasını indirdikten sonra istediğiniz bir konumda **readline** adında bir dizin oluşturun ve tar dosyasını oluşturulan dizin içinde açınız.

Paket adında(readline) 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
./build
```

Paket Derleme

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("Adınızı 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.

Paket Derleme

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 işlemlere 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="libc"
description="ncurses kütüphanesi"
source="https://ftp.gnu.org/pub/gnu/ncurses/${name}-${version}.tar.gz"
groups="sys.libs"
ROOTBUILDDIR="$HOME/distro/build"
BUILDDIR="$HOME/distro/build/build-${name}-${version}" #Derleme yapılan dizin
DESTDIR="$HOME/distro/rootfs" #Paketin yükleneceği sistem konumu
PACKAGEDIR=$(pwd)
SOURCEDIR="$HOME/distro/build/${name}-${version}"
initsetup(){
    mkdir -p $ROOTBUILDDIR #derleme dizini yoksa oluşturuluyor
    rm -rf $ROOTBUILDDIR/* #içeriği temizleniyor
    cd $ROOTBUILDDIR #dizinine geçiyoruz
    wget ${source}
    downloadfile=$(ls|head -1)
    filetype=$(file -b --extension $downloadfile|cut -d'/' -f1)
    if [ "${filetype}" == "???" ]; then unzip ${downloadfile}; else tar -xvf ${downloadfile};fi
    director=$(find ./ -maxdepth 0 -type d)
    if [ "${director}" != "${name}-${version}" ]; then mv $director ${name}-${version};fi
    cd $BUILDDIR
}

setup()
{
    ../${name}-${version}/configure \
        --prefix=/usr \
        --libdir=/lib64 \
        --with-shared \
        --disable-tic-depends \
        --with-versioned-syms \
        --enable-widex \
        --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
}
```

```
    # --without-ada
}
build()
{
    make
}
```

Paket Derleme

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

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
./build
```

Paket Derleme

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 oluşturmaya 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"
ROOTBUILDDIR="$HOME/distro/build"
BUILDDIR="$HOME/distro/build/build-${name}-${version}" #Derleme yapılan dizin
DESTDIR="$HOME/distro/rootfs" #Paketin yükleneceği sistem konumu
PACKAGEDIR=$(pwd)
SOURCEDIR="$HOME/distro/build/${name}-${version}"
initsetup(){
    mkdir -p $ROOTBUILDDIR #derleme dizini yoksa oluşturuluyor
    rm -rf $ROOTBUILDDIR/* #içeriği temizleniyor
    cd $ROOTBUILDDIR #dizinine geçiyoruz
    wget ${source}
    downloadfile=$(ls|head -1)
    filetype=$(file -b --extension $downloadfile|cut -d'/' -f1)
    if [ "${filetype}" == "???" ]; then unzip ${downloadfile}; else tar -xvf ${downloadfile};fi
    director=$(find ./ -maxdepth 0 -type d)
    if [ "${director}" != "${name}-${version}" ]; then mv $director ${name}-${version};fi
    cd $BUILDDIR
}

setup()
{
    cd $SOURCEDIR
    ./configure --prefix=/usr \
        --libdir=/usr/lib64 \
        --bindir=/bin \
        --with-curses \
        --enable-readline \
        --without-bash-malloc
}
build()
{
    make
}
package()
{
    make install DESTDIR=$DESTDIR
    cd $DESTDIR/bin
    ln -s bash sh
}

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(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.

Paket Derleme

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

Paket Derleme

openssl

coreutils için gerekli olan paket.

OpenSSL, açık kaynaklı bir kriptografik kütüphanedir ve genellikle ağ iletişimi güvenliği için kullanılır. SSL/TLS protokollerini uygulamak, şifreleme, dijital sertifikalar oluşturma ve doğrulama gibi işlemleri gerçekleştirmek için yaygın olarak tercih edilir. Özellikle web sunucuları, e-posta sunucuları ve diğer ağ uygulamaları için güvenli iletişim sağlamak amacıyla kullanılır. OpenSSL, Linux sistemlerinde sıkça kullanılan bir araçtır ve güvenli veri iletimi için önemli bir rol oynar.

Derleme

```
#!/usr/bin/env bash
version="3.2.0"
name="openssl"
depends="glibc,zlib"
description="openssl"
source="https://www.openssl.org/source/${name}-${version}.tar.gz"
groups="dev.libs"
BUILDDIR="$HOME/distro/build" #Derleme yapılan dizin
DESTDIR="$HOME/distro/rootfs" #Paketin yükleneceği sistem konumu
PACKAGEDIR=$(pwd)
SOURCEDIR="$BUILDDIR/${name}-${version}"
initsetup(){
    mkdir -p $BUILDDIR #derleme dizini yoksa oluşturuluyor
    rm -rf $BUILDDIR/* #içeriği temizleniyor
    cd $BUILDDIR #dizinine geçiyoruz
    wget ${source}
    downloadfile=$(ls|head -1)
    filetype=$(file -b --extension $downloadfile|cut -d'/' -f1)
    if [ "${filetype}" == "???" ]; then unzip ${downloadfile}; else tar -xvf ${downloadfile};fi
    director=$(find ./ -maxdepth 0 -type d)
    mv $director ${name}-${version};
}

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

build()
{
    make depend
    make
}

package()
{
    mkdir -p "${DESTDIR}/etc/ssl/" "${DESTDIR}/sbin/"
    install $SOURCEDIR/update-certdata "${DESTDIR}/sbin/update-certdata"
    install $SOURCEDIR/cacert.pem "${DESTDIR}/etc/ssl/cert.pem"
    make DESTDIR="${DESTDIR}" \
        install_sw \
        install_ssldirs \
        install_man_docs $jobs
}

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.
```

Yukarıdaki kodların sorunsuz çalışabilmesi için ek dosyalara ihtiyaç vardır. Bu ek dosyaları indirmek için [tıklayınız](#).

Paket Derleme

tar dosyasını indirdikten sonra istediğiniz bir konumda **openssl** adında bir dizin oluşturun ve tar dosyasını oluşturulan dizin içinde açınız.

Paket adında(openssl) 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  
./build
```


Paket Derleme

acl

coreutils için gerekli olan paket.

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 yapılabilir. Linux sistemlerinde, ACL'leri yönetmek için setfacl ve getfacl gibi komutlar kullanılır. Bu komutlar sayesinde dosya veya dizinlerin ACL yapılandırılmaları kolayca düzenlenebilir ve denetlenebilir.

Derleme

```
#!/usr/bin/env bash
name="acl"
version="2.3.1"
description="The acl package contains utilities to administer Access Control Lists"
source="https://download.savannah.nongnu.org/releases/acl/acl-${version}.tar.xz"
depends="attr"
group="sys.apps"
BUILDDIR="$HOME/distro/build" #Derleme yapılan dizin
DESTDIR="$HOME/distro/rootfs" #Paketin yükleneceği sistem konumu
PACKAGEDIR=$(pwd)
SOURCEDIR="$BUILDDIR/${name}-${version}"
initsetup(){
    mkdir -p $BUILDDIR #derleme dizini yoksa oluşturuluyor
    rm -rf $BUILDDIR/* #içeriği temizleniyor
    cd $BUILDDIR #dizinine geçiyoruz
    wget ${source}
    downloadfile=$(ls|head -1)
    filetype=$(file -b --extension $downloadfile|cut -d'/' -f1)
    if [ "${filetype}" == "???" ]; then unzip ${downloadfile}; else tar -xvf ${downloadfile};fi
    director=$(find ./ -maxdepth 0 -type d)
    mv $director ${name}-${version};
}
setup(){
    $SOURCEDIR/configure --prefix=/usr \
        --libdir=/usr/lib64
}
build(){
    make
}
package(){
    make install DESTDIR=$DESTDIR
}

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

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
./build
```

Paket Derleme

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

```
#!/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-${version}.tar.gz"
depends=""
group="sys.apps"
BUILDDIR="$HOME/distro/build" #Derleme yapılan dizin
DESTDIR="$HOME/distro/rootfs" #Paketin yükleneceği sistem konumu
PACKAGEDIR=$(pwd)
SOURCEDIR="$BUILDDIR/${name}-${version}"

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

setup(){
    $SOURCEDIR/configure --prefix=/usr \
        --sysconfdir=/etc \
        --libdir=/usr/lib64
}

build(){
    make
}

package(){
    make install DESTDIR=$DESTDIR
}

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

Paket 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
./build
```

Paket Derleme

libcap

coreutils için gerekli olan paket.

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.

Paket Derleme

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"
BUILDDIR="$HOME/distro/build" #Derleme yapılan dizin
DESTDIR="$HOME/distro/rootfs" #Paketin yükleneceği sistem konumu
PACKAGEDIR=$(pwd)
SOURCEDIR="$BUILDDIR/${name}-${version}"

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

_common_make_options=(
    CGO_CPPFLAGS="$CPPFLAGS"
    CGO_CFLAGS="$CFLAGS"
    CGO_CXXFLAGS="$CXXFLAGS"
    CGO_LDFLAGS="$LDFLAGS"
    CGO_REQUIRED="1"
    GOFLAGS="-buildmode=pie -mod=readonly -modcacherw"
    GO_BUILD_FLAGS="-ldflags '-compressdwarf=false -linkmode=external'"
)

setup()
{
    cd $SOURCEDIR
    cap_opts=(
        "${_common_make_options[@]}"
        SUDO=""
        prefix=/usr
        KERNEL_HEADERS=include
        lib=lib64
        sbindir=bin
        RAISE_SETFCAP=no
        #$(use_opt pam PAM_CAP=yes PAM_CAP=no)
    )
}

build()
{
    cd $SOURCEDIR
    make ${cap_opts[@]} DYNAMIC=yes
}

package()
{
    make "${_common_make_options[@]}" \
        DESTDIR="$DESTDIR" \
        RAISE_SETFCAP=no \
        prefix=/usr \
        lib=lib64 \
        sbindir=bin \
        install

    #mv $DESTDIR/lib64 $DESTDIR/usr/lib64
}

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

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

Paket Derleme

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

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

Paket Derleme

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${version}.zip"
groups="sys.libs"
BUILDDIR="$HOME/distro/build" #Derleme yapılan dizin
DESTDIR="$HOME/distro/rootfs" #Paketin yükleneceği sistem konumu
PACKAGEDIR=$(pwd)
SOURCEDIR="$BUILDDIR/${name}-${version}"

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

setup()
{
    cd $SOURCEDIR
    autoreconf -fvi
    cd $BUILDDIR
    ../${name}-${version}/configure --prefix=/usr \
        --libdir=/lib64 \
        $(use_opt python --with-python --without-python) \
        $(use_opt python3 --with-python3 --without-python3)
}

build()
{
    make
}

package()
{
    make install DESTDIR=$DESTDIR
}

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

Paket 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
./build
```

Paket Derleme

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-${version}.tar.xz"
groups="dev.libs"
BUILDDIR="$HOME/distro/build" #Derleme yapılan dizin
DESTDIR="$HOME/distro/rootfs" #Paketin yükleneceği sistem konumu
PACKAGEDIR=$(pwd)
SOURCEDIR="$BUILDDIR/${name}-${version}"

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

setup()
{
    cd $SOURCEDIR
    ./configure --prefix=/usr \
        --libdir=/usr/lib64 \
        $(use_opt cxx --enable-cxx --disable-cxx) \
        --enable-fat
}

build()
{
    make
}

package()
{
    make install DESTDIR=${DESTDIR}
}

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

Paket 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
./build
```

Paket Derleme

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-$version.tar.xz"
depends="libpcre2"
group="sys.apps"
BUILDDIR="$HOME/distro/build" #Derleme yapılan dizin
DESTDIR="$HOME/distro/rootfs" #Paketin yükleneceği sistem konumu
PACKAGEDIR=$(pwd)
SOURCEDIR="$BUILDDIR/${name}-${version}"

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

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

build(){
    make
}

package(){
    make install DESTDIR=$DESTDIR
}

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

Paket 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
./build
```


Paket Derleme

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-${version}.tar.xz"
depends="acl"
group="sys.apps"
BUILDDIR="$HOME/distro/build" #Derleme yapılan dizin
DESTDIR="$HOME/distro/rootfs" #Paketin yükleneceği sistem konumu
PACKAGEDIR=$(pwd)
SOURCEDIR="$BUILDDIR/${name}-${version}"

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

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

build(){
    make
}

package(){
    make install DESTDIR=$DESTDIR
}

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

Paket 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
./build
```

Paket Derleme

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-${version}.tar.xz"
groups="dev.libs"
BUILDDIR="$HOME/distro/build" #Derleme yapılan dizin
DESTDIR="$HOME/distro/rootfs" #Paketin yükleneceği sistem konumu
PACKAGEDIR=$(pwd)
SOURCEDIR="$BUILDDIR/${name}-${version}"

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

setup()
{
    cd $SOURCEDIR
    ./configure --prefix=/usr \
        --libdir=/usr/lib64 \
        --enable-shared \
        --enable-static \
        --disable-maintainer-mode \
        --enable-thread-safe
}

build()
{
    make
}

package()
{
    make install DESTDIR=${DESTDIR}
}

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

Paket 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
./build
```

Paket Derleme

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-$version.tar.xz"
depends="mpfr,readline"
group="sys.apps"
BUILDDIR="$HOME/distro/build" #Derleme yapılan dizin
DESTDIR="$HOME/distro/rootfs" #Paketin yükleneceği sistem konumu
PACKAGEDIR=$(pwd)
SOURCEDIR="$BUILDDIR/${name}-${version}"

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

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

build(){
    make
}

package(){
    make install DESTDIR=$DESTDIR
}

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

Paket 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
./build
```

Paket Derleme

findutils

Findutils paketi, Linux işletim sistemlerinde dosya ve izin 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-${version}.tar.xz"
depends=""
group="sys.apps"
BUILDDIR="$HOME/distro/build" #Derleme yapılan dizin
DESTDIR="$HOME/distro/rootfs" #Paketin yükleneceği sistem konumu
PACKAGEDIR=$(pwd)
SOURCEDIR="$BUILDDIR/${name}-${version}"

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

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

build(){
    make
}

package(){
    make install DESTDIR=$DESTDIR
}

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

Paket 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
./build
```

Paket Derleme

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"
BUILDDIR="$HOME/distro/build" #Derleme yapılan dizin
DESTDIR="$HOME/distro/rootfs" #Paketin yükleneceği sistem konumu
PACKAGEDIR=$(pwd)
SOURCEDIR="$BUILDDIR/${name}-${version}"

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

setup()
{
    ../${name}-${version}/configure --prefix=/usr \
        --libdir=/lib64 \
        --enable-shared \
        --enable-static \
        --enable-pcre2-16 \
        --enable-pcre2-32 \
        --enable-jit \
        --enable-pcre2test-libreadline
}

build()
{
    make
}

package()
{
    make install DESTDIR=$DESTDIR
}

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

Paket 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
./build
```

Paket Derleme

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-$version.tar.xz"
depends="acl,attr,gmp,libcap,openssl"
group="sys.apps"
BUILDDIR="$HOME/distro/build" #Derleme yapılan dizin
DESTDIR="$HOME/distro/rootfs" #Paketin yükleneceği sistem konumu
PACKAGEDIR=$(pwd)
SOURCEDIR="$BUILDDIR/${name}-${version}"

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

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

build(){
    make
}

package(){
    make install DESTDIR=$DESTDIR
}

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

Paket 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
./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.

Paket Derleme

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=""
builddepend="libcap-ng,python,eudev,sqlite,eudev,cryptsetup,libxcrypt"
group="sys.apps"
BUILDDIR="$HOME/distro/build" #Derleme yapılan dizin
DESTDIR="$HOME/distro/rootfs" #Paketin yükleneceği sistem konumu
PACKAGEDIR=$(pwd)
SOURCEDIR="$BUILDDIR/${name}-${version}"

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

setup(){
    cp -prvf $PACKAGEDIR/files/ /tmp/bps/build/
    cd $SOURCEDIR
    patch -Np1 -i ../files/0001-util-linux-tmpfiles.patch

    ./configure --prefix=/usr \
        --libdir=/usr/lib64 \
        --bindir=/usr/bin \
        --enable-shared \
        --enable-static \
        --disable-su \
        --disable-runuser \
        --disable-chfn-chsh \
        --disable-login \
        --disable-sulogin \
        --disable-makeinstall-chown \
        --disable-makeinstall-setuid \
        --disable-pylibmount \
        --disable-raw \
        --without-systemd \
        --without-libuser \
        --without-utempter \
        --without-econf \
        --with-python \
        --with-udev
}

build(){
```

```
    make

}

package(){
    _python_stdlib="$(python -c 'import sysconfig; print(sysconfig.get_paths()["stdlib"])' )"
    make install DESTDIR=$DESTDIR

    # remove static libraries
    rm "${DESTDIR}"/usr/lib/lib*.a*

    # setuid chfn and chsh
    chmod 4755 "${DESTDIR}"/usr/bin/{newgrp,ch{sh,fn}}

    # install PAM files for login-utils
```

```
install -Dm0644 ../files/pam-common "${DESTDIR}/etc/pam.d/chfn"
install -m0644 ../files/pam-common "${DESTDIR}/etc/pam.d/chsh"
install -m0644 ../files/pam-login "${DESTDIR}/etc/pam.d/login"
```

Paket Derleme

```
install -m0644 ../files/pam-remote "${DESTDIR}/etc/pam.d/remote"
install -m0644 ../files/pam-runuser "${DESTDIR}/etc/pam.d/runuser"
install -m0644 ../files/pam-runuser "${DESTDIR}/etc/pam.d/runuser-l"
install -m0644 ../files/pam-su "${DESTDIR}/etc/pam.d/su"
install -m0644 ../files/pam-su "${DESTDIR}/etc/pam.d/su-l"

mkdir -p $DESTDIR/usr/lib64/python3.11

# runtime libs are shipped as part of util-linux-libs
install -d -m0755 util-linux-libs/lib/
mv "$DESTDIR"/usr/lib/lib*.so* util-linux-libs/lib64/
mv "$DESTDIR"/usr/lib/pkgconfig util-linux-libs/lib64/pkgconfig
mv "$DESTDIR"/usr/include util-linux-libs/include
mv "$DESTDIR"/"${_python_stdlib}"/site-packages util-linux-libs/site-packages
rmdir "$DESTDIR"/"${_python_stdlib}"
mv "$DESTDIR"/usr/share/man/man3 util-linux-libs/man3

mv util-linux-libs/lib/* "$DESTDIR"/usr/lib64/
mv util-linux-libs/include "$DESTDIR"/usr/include
mv util-linux-libs/site-packages "$DESTDIR"/"${_python_stdlib}"/site-packages

# install esysusers
install -Dm0644 ../files/util-linux.sysusers "${DESTDIR}/usr/lib64/sysusers.d/util-linux.conf"

install -Dm0644 ../files/60-rfkill.rules "${DESTDIR}/usr/lib64/udev/rules.d/60-rfkill.rules"
}
initsetup      # initsetup fonksiyonunu çalıştırır ve kaynak dosyayı indirir
setup          # setup fonksiyonu çalışır ve derleme öncesi kaynak dosyaların ayarlanması sağlanır.
build          # build fonksiyonu çalışır ve kaynak dosyaları derlenir.
package        # package fonksiyonu çalışır, yükleme öncesi ayarlamalar yapılır ve yüklenir.
```

Yukarıdaki kodların sorunsuz çalışabilmesi için ek dosyalara 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 açınız.

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
./build
```

Paket Derleme

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"
BUILDDIR="$HOME/distro/build" #Derleme yapılan dizin
DESTDIR="$HOME/distro/rootfs" #Paketin yükleneceği sistem konumu
PACKAGEDIR=$(pwd)
SOURCEDIR="$BUILDDIR/${name}-${version}"

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

setup()
{
    export DEFS="NO_ASM"
    cd $SOURCEDIR
    autoreconf -fvi
    ./configure --prefix=/usr \
    --libdir=/usr/lib64/
}

build()
{
    make
}

package()
{
    make install DESTDIR=$DESTDIR
}

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

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
./build
```

Paket Derleme

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-${version}.tar.gz"
md5sums="66f82a9fa24623f5ea8a9ee6b4f808e2"
group="app.arch"
BUILDDIR="$HOME/distro/build" #Derleme yapılan dizin
DESTDIR="$HOME/distro/rootfs" #Paketin yükleneceği sistem konumu
PACKAGEDIR=$(pwd)
SOURCEDIR="$BUILDDIR/${name}-${version}"

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

setup(){
    $SOURCEDIR/configure --prefix=/usr \
        --libdir=/usr/lib64 \
        --enable-static \
        --enable-shared \
        --enable-doc \
        --enable-nls
    #$(use_opt doc --enable-doc --disable-doc) \
    #$(use_opt nls --enable-nls --disable-nls)
}

build(){
    make $jobs
}

package(){
    make install DESTDIR=$DESTDIR
}

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

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
./build
```

Paket Derleme

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"
BUILDDIR="$HOME/distro/build" #Derleme yapılan dizin
DESTDIR="$HOME/distro/rootfs" #Paketin yükleneceği sistem konumu
PACKAGEDIR=$(pwd)
SOURCEDIR="$BUILDDIR/${name}-${version}"

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

setup()
{
    cd $SOURCEDIR
}

build()
{
    make
}

package()
{
    make prefix=/ install DESTDIR=$DESTDIR
}

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

Paket 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
./build
```

bzip2

bzip2, dosyaları sıkıştırmak için kullanılan bir yazılımdır ve genellikle Unix tabanlı sistemlerde tercih edilmektedir. Bu araç, dosyaların boyutunu önemli ölçüde azaltarak depolama alanından tasarruf sağlar ve veri transferini hızlandırır. bzip2, Lempel-Ziv ve Burrows-Wheeler algoritmalarını kullanarak yüksek sıkıştırma oranları suna

Paket Derleme

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"
BUILDDIR="$HOME/distro/build" #Derleme yapılan dizin
DESTDIR="$HOME/distro/rootfs" #Paketin yükleneceği sistem konumu
PACKAGEDIR=$(pwd)
SOURCEDIR="$BUILDDIR/${name}-${version}"

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

setup()
{
    cd $SOURCEDIR

# Generate relative symlinks
    sed -i \
        -e 's:\$(PREFIX)/man:\$(PREFIX)/share/man:g' \
        -e 's:ln -s -f $(PREFIX)/bin:ln -s : ' \
        Makefile

# fixup broken version stuff
    sed -i \
        -e "s:1\.\0\.\4:$version:" \
        bzip2.1 bzip2.txt Makefile-libbz2_so manual.*
}

build()
{
    make -f Makefile-libbz2_so all
    make all
}

package()
{
    cd $SOURCEDIR
    make PREFIX="$DESTDIR"/usr install
    install -D libbz2.so.$version "$DESTDIR"/usr/lib/libbz2.so.$pkgver
    ln -s libbz2.so.$version "$DESTDIR"/usr/lib/libbz2.so
    ln -s libbz2.so.$version "$DESTDIR"/usr/lib/libbz2.so.${version%.*}

    mkdir -p "$DESTDIR"/usr/lib/pkgconfig/
    sed "s|@VERSION@|$version|" $SOURCEDIR/bzip2.pc.in > "$DESTDIR"/usr/lib/pkgconfig/bzip2.pc
}

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(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.

Paket Derleme

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


Paket Derleme

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"
BUILDDIR="$HOME/distro/build" #Derleme yapılan dizin
DESTDIR="$HOME/distro/rootfs" #Paketin yükleneceği sistem konumu
PACKAGEDIR=$(pwd)
SOURCEDIR="$BUILDDIR/${name}-${version}"

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

setup(){
    cd $SOURCEDIR
    ./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(){
    pwd
    make
}

package(){
    make install DESTDIR=$DESTDIR
}

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

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

Paket Derleme

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

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

Paket Derleme

libselinux

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

Derleme

```
#!/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"
BUILDDIR="$HOME/distro/build" #Derleme yapılan dizin
DESTDIR="$HOME/distro/rootfs" #Paketin yükleneceği sistem konumu
PACKAGEDIR=$(pwd)
SOURCEDIR="$BUILDDIR/${name}-${version}"

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

setup()
{
    echo ""
}

build()
{
    cp -prvf $PACKAGEDIR/files/lfs64.patch /tmp/bps/build/
    cd $SOURCEDIR
    patch -Np1 -i ../lfs64.patch
    make FTS_LDLIBS="-lfts"
}

package()
{
    cd $SOURCEDIR
    make install DESTDIR=$DESTDIR
    #mv ${DESTDIR}/lib ${DESTDIR}/lib64
}

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

Yukarıdaki kodların sorunsuz çalışabilmesi için ek dosyalara 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 açınız.

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.

Paket Derleme

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

Paket Derleme

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

```
#!/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"
BUILDDIR="$HOME/distro/build" #Derleme yapılan dizin
DESTDIR="$HOME/distro/rootfs" #Paketin yükleneceği sistem konumu
PACKAGEDIR=$(pwd)
SOURCEDIR="$BUILDDIR/${name}-${version}"

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

setup()
{
    echo ""
}

build()
{
    cd $SOURCEDIR
    make
}

package()
{
    cd $SOURCEDIR
    make install DESTDIR=$DESTDIR
}

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

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
./build
```

Paket Derleme

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-$version.tar.xz"
depends="glibc,acl"
builddepend=""
groups="app.arch"
BUILDDIR="$HOME/distro/build" #Derleme yapılan dizin
DESTDIR="$HOME/distro/rootfs" #Paketin yükleneceği sistem konumu
PACKAGEDIR=$(pwd)
SOURCEDIR="$BUILDDIR/${name}-${version}"

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

setup()
{
    cd $SOURCEDIR
    ./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
}

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

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
./build
```

Paket Derleme

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$version.tar.gz"
groups="app.arch"
BUILDDIR="$HOME/distro/build" #Derleme yapılan dizin
DESTDIR="$HOME/distro/rootfs" #Paketin yükleneceği sistem konumu
PACKAGEDIR=$(pwd)
SOURCEDIR="$BUILDDIR/${name}-${version}"

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

setup()
{
    cd $SOURCEDIR
    ./configure --prefix=/usr
}

build()
{
    make
}

package()
{
    make install pkgconfigdir="/usr/lib64/pkgconfig" DESTDIR=$DESTDIR
}

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

Paket 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
./build
```

Paket Derleme

brothli

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

```
#!/usr/bin/env bash
version="1.1.0"
name="brothli"
depends="glibc,zlib"
description="Generic-purpose lossless compression algorithm"
source="https://github.com/google/brotli/archive/refs/tags/v$version.tar.gz"
groups="sys.apps"
BUILDDIR="$HOME/distro/build" #Derleme yapılan dizin
DESTDIR="$HOME/distro/rootfs" #Paketin yükleneceği sistem konumu
PACKAGEDIR=$(pwd)
SOURCEDIR="$BUILDDIR/${name}-${version}"

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

setup()
{
    cmake ../${name}-${version} \
        -DCMAKE_INSTALL_PREFIX=/usr \
        -DBUILD_SHARED_LIBS=True
}

build()
{
    make
}

package()
{
    make install DESTDIR=$DESTDIR
    mkdir -p $DESTDIR/lib
    cp -prfv $DESTDIR/usr/lib/x86_64-linux-gnu/* $DESTDIR/lib/
}

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

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
./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.

Paket Derleme

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"
BUILDDIR="$HOME/distro/build" #Derleme yapılan dizin
DESTDIR="$HOME/distro/rootfs" #Paketin yükleneceği sistem konumu
PACKAGEDIR=$(pwd)
SOURCEDIR="$BUILDDIR/${name}-${version}"

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

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-resolver
        --enable-websockets
        #$(use_opt zstd --with-zstd --without-zstd)
        #$(use_opt zlib --with-zlib --without-zlib)
        #$(use_opt brotli --with-brotli --without-brotli)
        #$(use_opt libssh --with-libssh --without-libssh)
        #$(use_opt libidn2 --with-libidn2 --without-libidn2)
    )

    cd $SOURCEDIR
    ./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
}

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

Paket Derleme

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  
./build
```

Paket Derleme

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${version}.tar.gz"
groups="dev.libs"
BUILDDIR="$HOME/distro/build" #Derleme yapılan dizin
DESTDIR="$HOME/distro/rootfs" #Paketin yükleneceği sistem konumu
PACKAGEDIR=$(pwd)
SOURCEDIR="$BUILDDIR/${name}-${version}"

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

setup()
{
    cd $SOURCEDIR
    autoreconf -fvi
    #NOCONFIGURE=1 ./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/
    rm -rf $DESTDIR/usr/lib
}

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

Paket Derleme

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  
./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. Eudev, özellikle daha hafif ve daha az bağımlılığa sahip bir alternatif arayan kullanıcılar için geliştirilmiştir.

Paket Derleme

Derleme

```
#!/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"
BUILDDIR="$HOME/distro/build" #Derleme yapılan dizin
DESTDIR="$HOME/distro/rootfs" #Paketin yükleneceği sistem konumu
PACKAGEDIR=$(pwd)
SOURCEDIR="$BUILDDIR/${name}-${version}"

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

setup()
{
    cp $PACKAGEDIR/files/eudev.hook $SOURCEDIR
    cp $PACKAGEDIR/files/eudev.init-bottom $SOURCEDIR
    cp $PACKAGEDIR/files/eudev.init-top $SOURCEDIR

    $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 \
        #--enable-blkid \
}

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/udevdev udevdev
    mkdir -p ${DESTDIR}/usr/lib64/pkgconfig/
    cd ${DESTDIR}/usr/lib64/pkgconfig/
    ln -s ../../lib64/pkgconfig/libudev.pc libudev.pc
    #ln -sv libudev.pc "$DESTDIR/usr/lib64/pkgconfig/libudev.pc"
}

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

Paket Derleme

Yukarıdaki kodların sorunsuz çalışabilmesi için ek dosyalara 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 açınız.

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  
./build
```


Paket Derleme

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"
BUILDDIR="$HOME/distro/build" #Derleme yapılan dizin
DESTDIR="$HOME/distro/rootfs" #Paketin yükleneceği sistem konumu
PACKAGEDIR=$(pwd)
SOURCEDIR="$BUILDDIR/${name}-${version}"

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

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

build(){
    make $jobs
}

package(){
    make install DESTDIR=$DESTDIR
}

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

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
./build
```

Paket Derleme

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"
BUILDDIR="$HOME/distro/build" #Derleme yapılan dizin
DESTDIR="$HOME/distro/rootfs" #Paketin yükleneceği sistem konumu
PACKAGEDIR=$(pwd)
SOURCEDIR="$BUILDDIR/${name}-${version}"

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

setup()
{
    ../${name}-${version}/configure --prefix=/usr \
        --sbindir=/usr/sbin \
        --libdir=/usr/lib64 \
        --enable-securedir=/usr/lib64/security \
        --enable-static \
        --enable-shared \
        --disable-nls \
        --disable-selinux \
        --disable-db
}

build()
{
    make
}

package()
{
    make install DESTDIR=$DESTDIR
    chmod +s "$DESTDIR"/usr/sbin/unix_chkpwd
}

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

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.

Paket Derleme

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

Paket Derleme

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"
vrnsn="2_6_2"
description="An XML parser library"
#source="https://github.com/libexpat/libexpat/archive/refs/tags/R_${version}.tar.gz"
source="https://github.com/libexpat/libexpat/releases/download/R_${vrnsn}/expat-${version}.tar.bz2"
depends=""
builddepend=""
group="dev.libs"

BUILDDIR="$HOME/distro/build" #Derleme yapılan dizin
DESTDIR="$HOME/distro/rootfs" #Paketin yükleneceği sistem konumu
PACKAGEDIR=$(pwd)
SOURCEDIR="$BUILDDIR/${name}-${version}"

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

setup(){
    cd $SOURCEDIR
    #cd ../$name-$version/$name
    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
}

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(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.

Paket Derleme

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

Paket Derleme

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-$version.tar.xz"
BUILDDIR="$HOME/distro/build" #Derleme yapılan dizin
DESTDIR="$HOME/distro/rootfs" #Paketin yükleneceği sistem konumu
PACKAGEDIR=$(pwd)
SOURCEDIR="$BUILDDIR/${name}-${version}"

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

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

build(){
    make
}

package(){
    make install DESTDIR=$DESTDIR
}

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

Paket 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
./build
```

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. Audit, özellikle güvenlik ihlallerinin tespit edilmesi ve sistemin uyumluluk gereksinimlerini karşılaması açısından kritik bir rol oynar.

Paket Derleme

Derleme

```
#!/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$version.tar.gz"
groups="sys.process"
BUILDDIR="$HOME/distro/build" #Derleme yapılan dizin
DESTDIR="$HOME/distro/rootfs" #Paketin yükleneceği sistem konumu
PACKAGEDIR=$(pwd)
SOURCEDIR="$BUILDDIR/${name}-${version}"

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

setup()
{
    cp $PACKAGEDIR/files/auditd.initd $SOURCEDIR/auditd.initd
    cp $PACKAGEDIR/files/auditd.conf.d $SOURCEDIR/auditd.conf.d

    cd $SOURCEDIR
    autoreconf -fv --install
    cd $BUILDDIR
    ../${name}-${version}/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 $SOURCEDIR/auditd.initd "$DESTDIR"/etc/init.d/auditd
    install -Dm755 $SOURCEDIR/auditd.conf.d "$DESTDIR"/etc/conf.d/auditd
}

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.
```

Yukarıdaki kodların sorunsuz çalışabilmesi için ek dosyalara ihtiyaç vardır. Bu ek dosyaları indirmek için [tıklayınız](#).

Paket Derleme

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

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  
./build
```

Paket Derleme

libgcc

libgcc 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.

Paket Derleme

Derleme

```
#!/usr/bin/env bash
version="13.1.0"
name="libgcc"
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"
BUILDDIR="$HOME/distro/build" #Derleme yapılan dizin
DESTDIR="$HOME/distro/rootfs" #Paketin yükleneceği sistem konumu
PACKAGEDIR=$(pwd)
SOURCEDIR="$BUILDDIR/${name}-${version}"

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

export CFLAGS="-O2 -s"
export CXXFLAGS="-O2 -s"
unset LDFLAGS
setup()
{
case $(uname -m) in
x86_64)
sed -i.orig '/m64=s/lib64/lib/' gcc/config/i386/t-linux64
;;
esac

options=(
--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
```

Paket Derleme

```
--without-zstd
--disable-nls
)
cd $SOURCEDIR
mkdir build
cd build
../configure ${options[@]} \
  --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 $DESTDIR
    #find ./ -iname "*" -exec strip -s {} \;
    while read -rd '' file; do
    case "${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)

}

yedeke(){
    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 "." -type f -iname "*" -print0)

}

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

Paket adında(libgcc) 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
./build
```

Paket Derleme

libxcrypt

libxcrypt, Unix benzeri sistemlerde parolaların şifrlenmesi 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=""
BUILDDIR="$HOME/distro/build" #Derleme yapılan dizin
DESTDIR="$HOME/distro/rootfs" #Paketin yükleneceği sistem konumu
PACKAGEDIR=$(pwd)
SOURCEDIR="$BUILDDIR/${name}-${version}"

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

setup() {
    ../${name}-${version}/configure --prefix=/usr \
        --libdir=/usr/lib64/ \
        --sbindir=/usr/bin
}

build() {
    make -C $BUILDDIR
}

package() {
    make -C $BUILDDIR install DESTDIR=$DESTDIR
}

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

Paket 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
./build
```

Paket Derleme

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-$srcver.tar.gz"
depends="zlib,readline"
group="dev.db"
BUILDDIR="$HOME/distro/build" #Derleme yapılan dizin
DESTDIR="$HOME/distro/rootfs" #Paketin yükleneceği sistem konumu
PACKAGEDIR=$(pwd)
SOURCEDIR="$BUILDDIR/${name}-${version}"

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

setup(){
    #cd $name-autoconf-$srcver
    cd $SOURCEDIR
    ./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
}

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(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.

Paket Derleme

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

Paket Derleme

libtirpc

libtirpc, UNIX sistemlerinde yaygın olarak kullanılan bir RPC kütüphanesidir. Bu kütüphane, istemci ve sunucu uygulamaları arasında uzaktan prosedür çağrılarını 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-$version.tar.bz2"
depends=""
group="net.libs"
BUILDDIR="$HOME/distro/build" #Derleme yapılan dizin
DESTDIR="$HOME/distro/rootfs" #Paketin yükleneceği sistem konumu
PACKAGEDIR=$(pwd)
SOURCEDIR="$BUILDDIR/${name}-${version}"

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

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

build(){
    make
}

package(){
    make install DESTDIR=$DESTDIR
}

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

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
./build
```


Paket Derleme

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-${version}.tar.gz")
group=(sys.apps)
depends=""
BUILDDIR="$HOME/distro/build" #Derleme yapılan dizin
DESTDIR="$HOME/distro/rootfs" #Paketin yükleneceği sistem konumu
PACKAGEDIR=$(pwd)
SOURCEDIR="$BUILDDIR/${name}-${version}"

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

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

build(){
    make
}

package(){
    make install DESTDIR=$DESTDIR
}

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(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
./build
```


Paket Derleme

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}-${version}.tar.xz"
groups="sys.fs"
BUILDDIR="$HOME/distro/build" #Derleme yapılan dizin
DESTDIR="$HOME/distro/rootfs" #Paketin yükleneceği sistem konumu
PACKAGEDIR=$(pwd)
SOURCEDIR="$BUILDDIR/${name}-${version}"

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

setup()
{
    cd $SOURCEDIR
    ./configure --sbindir=/usr/bin \
        --libdir=/usr/lib64/
}

build()
{
    make
}

package()
{
    make install DESTDIR=$DESTDIR
    rm -rf $DESTDIR/usr/share/man/man8/fsck.8
}

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

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
./build
```

Paket Derleme

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$version.tar.gz"
groups="sys.block"
BUILDDIR="$HOME/distro/build" #Derleme yapılan dizin
DESTDIR="$HOME/distro/rootfs" #Paketin yükleneceği sistem konumu
PACKAGEDIR=$(pwd)
SOURCEDIR="$BUILDDIR/${name}-${version}"

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

setup()
{
    cd $SOURCEDIR
    ./autogen.sh
    ./configure --prefix=/usr \
        --libdir=/usr/lib64/ \
        --enable-compat-symlinks
}

build()
{
    cd $SOURCEDIR
    make
}

package()
{
    cd $SOURCEDIR
    make install DESTDIR=$DESTDIR
}

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

Paket 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
./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. Kullanıcı isterse kendi scriptinide kullanabilir. Debian dışında **dracut** aracıda initramfs oluşturmak ve güncellemek için kullanılabilir.

Paket Derleme

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"
BUILDDIR="$HOME/distro/build" #Derleme yapılan dizin
DESTDIR="$HOME/distro/rootfs" #Paketin yükleneceği sistem konumu
PACKAGEDIR=$(pwd)
SOURCEDIR="$BUILDDIR/${name}-${version}"

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

setup()
{
    mkdir -p $SOURCEDIR/conf-hooks.d
    cp ${dizin}/${paket}/conf-hooks.d/* $SOURCEDIR/conf-hooks.d/
    mkdir -p $SOURCEDIR/patches
    cp $PACKAGEDIR/files/patches/* $SOURCEDIR/patches/
    cp $PACKAGEDIR/files/initramfs-tools.sysconf $SOURCEDIR/initramfs-tools.sysconf
    cp $PACKAGEDIR/files/zzz-busybox $SOURCEDIR/zzz-busybox
    #cp $PACKAGEDIR/files/fsck $SOURCEDIR/fsck
    cp $PACKAGEDIR/files/modules $SOURCEDIR/modules

    cd $SOURCEDIR
    patch -Np1 < $SOURCEDIR/patches/remove-zstd.patch
    patch -Np1 < $SOURCEDIR/patches/remove-logsav.patch
    patch -Np1 < $SOURCEDIR/patches/non-debian.patch
}

build()
{
    echo ""
}

package()
{
    cd $SOURCEDIR
    cat debian/*.install | sed "s/\t/ /g" | tr -s " " | while read line ; do
        file=$(echo $line | cut -f1 -d" ")
        target=$(echo $line | cut -f2 -d" ")
        mkdir -p ${DESTDIR}/${target}
        cp -prvf $file ${DESTDIR}/${target}/
    done
}
```

```
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 ../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/
```

Paket Derleme

```
mkdir -p ${DESTDIR}/etc/initramfs-tools/scripts

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

Yukarıdaki kodların sorunsuz çalışabilmesi için ek dosyalara 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 açınız.

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
./build
```

/etc/initramfs-tools/modules

modules dosyası initrd oluşturulma ve güncelleme durumunda isteğe bağlı olarak modüllerin eklenmesini ve **initrd** açıldığında modülün yüklenmesini istiyorsak **/etc/initramfs-tools/modules** komundaki dosyayı aşağıdaki gibi düzenlemeliyiz. Bu dosya içinde **ext4**, **vfat** ve diğer yardımcı modüller 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.

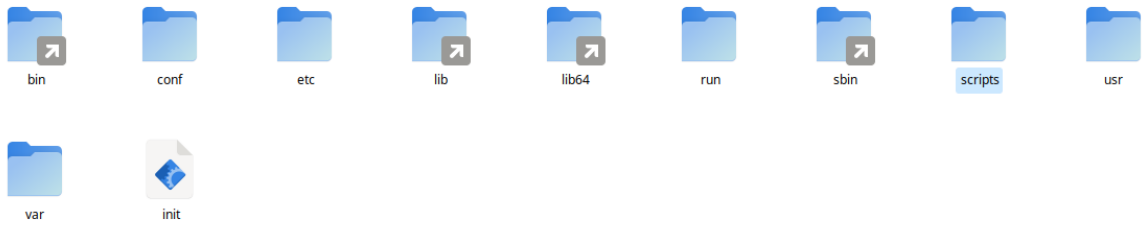
initramfs-tools Güncelleme

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

Güncelleme ve oluşturma aşamasında **/usr/share/initramfs-tools/hooks/** konumundaki dosyaları 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

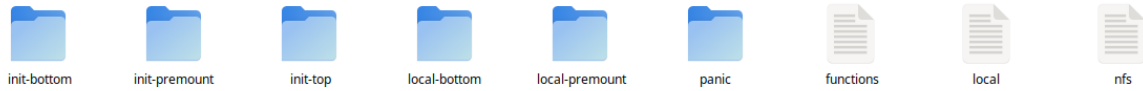
Sistemin açılması için **vmlinuz**, **initrd.img** ve **grub.cfg** dosyalarının olması yeterlidir. **initrd.img** sistemin açılma sürecini yürüten bir kernel yardımcı ön sistemidir. **initrd.img** açıldığında aşağıdaki gibi bir dizin yapısı olur. Bu dizinler içindeki **script** dizini çok önemlidir.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



initrd script İçeriği

script içindeki 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.

Paket Derleme

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-${version}.tar.gz"
depends="glibc"
group="app.arch"
BUILDDIR="$HOME/distro/build" #Derleme yapılan dizin
DESTDIR="$HOME/distro/rootfs" #Paketin yükleneceği sistem konumu
PACKAGEDIR=$(pwd)
SOURCEDIR="$BUILDDIR/${name}-${version}"

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

setup(){
    cd $SOURCEDIR
    CFLAGS+=' -fcommon'
    ./configure --prefix=/usr
}

build(){
    make
}

package(){
    make install DESTDIR=$DESTDIR
}

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

Paket 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
./build
```

Paket Derleme

libio

Libio, C dilinde girdi/çıkı 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"
BUILDDIR="$HOME/distro/build" #Derleme yapılan dizin
DESTDIR="$HOME/distro/rootfs" #Paketin yükleneceği sistem konumu
PACKAGEDIR=$(pwd)
SOURCEDIR="$BUILDDIR/${name}-${version}"

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

setup (){
    echo ""
}

build(){
    cd $SOURCEDIR
    make
}

package(){
    make install DESTDIR=$DESTDIR
}

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

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
./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.

Paket Derleme

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$version.tar.gz"
depends="libaio"
builddepend=""
group="sys.fs"
BUILDDIR="$HOME/distro/build" #Derleme yapılan dizin
DESTDIR="$HOME/distro/rootfs" #Paketin yükleneceği sistem konumu
PACKAGEDIR=$(pwd)
SOURCEDIR="$BUILDDIR/${name}-${version}"

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

setup(){
    cd $SOURCEDIR
    ./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
}

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

Paket Derleme

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

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

Paket Derleme

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 cmdline option parser"
source="http://ftp.rpm.org/popt/releases/popt-${version%.*}.x/popt-${version}.tar.gz"
depends=""
builddepend=""
group="sys.apps"
BUILDDIR="$HOME/distro/build" #Derleme yapılan dizin
DESTDIR="$HOME/distro/rootfs" #Paketin yükleneceği sistem konumu
PACKAGEDIR=$(pwd)
SOURCEDIR="$BUILDDIR/${name}-${version}"

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

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

build(){
    make
}

package(){
    make install DESTDIR=$DESTDIR
}

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

Paket 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
./build
```

Paket Derleme

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)

BUILDDIR="$HOME/distro/build" #Derleme yapılan dizin
DESTDIR="$HOME/distro/rootfs" #Paketin yükleneceği sistem konumu
PACKAGEDIR=$(pwd)
SOURCEDIR="$BUILDDIR/${name}-${version}"

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

setup(){
    cd $SOURCEDIR/source
    #autoreconf -fvi
    ./configure --prefix=/usr \
        --libdir=/usr/lib64/
}

build(){
    make
}

package(){
    make install DESTDIR=$DESTDIR
    chmod +x "$DESTDIR"/usr/bin/icu-config
}

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

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
./build
```

Paket Derleme

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"
BUILDDIR="$HOME/distro/build" #Derleme yapılan dizin
DESTDIR="$HOME/distro/rootfs" #Paketin yükleneceği sistem konumu
PACKAGEDIR=$(pwd)
SOURCEDIR="$BUILDDIR/${name}-${version}"

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

setup(){
    cd "${SOURCEDIR}"
    cp ${PACKAGEDIR}/files/* $SOURCEDIR/
    # set correct fhs structure
    patch -Np1 -i "${SOURCEDIR}"/0001-make-iproute2-fhs-compliant.patch

    # use Berkeley DB 5.3
    patch -Np1 -i "${SOURCEDIR}"/0002-bdb-5-3.patch

    # do not treat warnings as errors
    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
}

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.
```

Yukarıdaki kodların sorunsuz çalışabilmesi için ek dosyalara ihtiyaç vardır. Bu ek dosyaları indirmek için [tıklayınız](#).

Paket Derleme

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

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  
./build
```

Paket Derleme

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"
BUILDDIR="$HOME/distro/build" #Derleme yapılan dizin
DESTDIR="$HOME/distro/rootfs" #Paketin yükleneceği sistem konumu
PACKAGEDIR=$(pwd)
SOURCEDIR="$BUILDDIR/${name}-${version}"

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

setup(){
    export BINDIR="/usr/bin" SBINDIR="/usr/bin"
    cd $SOURCEDIR/
    #${SOURCEDIR}/configure --prefix=/usr \
    #    --libdir=/usr/lib64/
    echo ""
}

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
}

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(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
./build
```


Paket Derleme

dhcp

DHCP, dinamik IP adresi atama sürecini otomatikleştirerek ağ yöneticilerinin iş yükünü azaltır. 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"
BUILDDIR="$HOME/distro/build" #Derleme yapılan dizin
DESTDIR="$HOME/distro/rootfs" #Paketin yükleneceği sistem konumu
PACKAGEDIR=$(pwd)
SOURCEDIR="$BUILDDIR/${name}-${version}"

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

setup(){
    $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 $PACKAGEDIR/dhclient.init.d $DESTDIR/etc/init.d/dhclient
    install $PACKAGEDIR/dhclient.init.d ${DESTDIR}/etc/runlevels/default/dhclient
}

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.
```

Yukarıdaki kodların sorunsuz çalışabilmesi için ek dosyalara 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 açınız.

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

Paket Derleme

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

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

Paket Derleme

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.

Paket Derleme

Derleme

```
#!/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"
BUILDDIR="$HOME/distro/build" #Derleme yapılan dizin
DESTDIR="$HOME/distro/rootfs" #Paketin yükleneceği sistem konumu
PACKAGEDIR=$(pwd)
SOURCEDIR="$BUILDDIR/${name}-${version}"

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

setup(){
    mkdir -p /tmp/bps/build/files
    cp -prvf $PACKAGEDIR/files/ /tmp/bps/build/
    cd $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=$DESTDIR
    # remove selinux stuff
    mkdir -p "${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 ${DESTDIR}/etc/group ]; then
    echo -e "root:x:0:">${DESTDIR}/etc/group
    echo -e "users:x:2:">${DESTDIR}/etc/group
    echo -e "tty:x:5:">${DESTDIR}/etc/group
    echo -e "disk:x:6:">${DESTDIR}/etc/group
    echo -e "dialout:x:20:">${DESTDIR}/etc/group
    echo -e "cdrom:x:24:">${DESTDIR}/etc/group
    echo -e "audio:x:29:">${DESTDIR}/etc/group
fi
```

Paket Derleme

```
echo -e "video:x:44:">>${DESTDIR}/etc/group
echo -e "wheel:x:120:">>${DESTDIR}/etc/group
echo -e "plugdev:x:121:">>${DESTDIR}/etc/group
echo -e "netdev:x:122:">>${DESTDIR}/etc/group
echo -e "dip:x:123:">>${DESTDIR}/etc/group

chmod 644 ${DESTDIR}/etc/group
chown root ${DESTDIR}/etc/group
chgrp root ${DESTDIR}/etc/group
else
chmod 644 ${DESTDIR}/etc/group
chown root ${DESTDIR}/etc/group
chgrp root ${DESTDIR}/etc/group
fi

if [ ! -f ${DESTDIR}/etc/shadow ] ; then
echo "root:*::0::::::" > ${DESTDIR}/etc/shadow
chmod 600 ${DESTDIR}/etc/shadow
chown root ${DESTDIR}/etc/shadow
chgrp root ${DESTDIR}/etc/shadow
else
chmod 600 ${DESTDIR}/etc/shadow
chown root ${DESTDIR}/etc/shadow
chgrp root ${DESTDIR}/etc/shadow
fi

if [ ! -f "${DESTDIR}/etc/passwd" ]; then
echo -e "root:x:0:0:root:/root:/bin/sh">${DESTDIR}/etc/passwd
fi

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

Yukarıdaki kodların sorunsuz çalışabilmesi için ek dosyalara 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 açınız.

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
./build
```


openrc

OpenRC, sistem başlangıcını ve hizmetlerin yönetimini sağlamak amacıyla geliştirilmiş bir init sistemidir. Gentoo Linux gibi dağıtımlarda yaygın olarak kullanılmakta olup, sistemin daha esnek ve modüler bir şekilde yönetilmesine olanak tanır. OpenRC, sistem hizmetlerini başlatmak, durdurmak ve yeniden başlatmak için bir dizi betik ve yapılandırma dosyası kullanır.

Paket Derleme

Derleme

```
#!/usr/bin/env bash
name="openrc"
version="0.53"
description="The OpenRC init system"
source="https://github.com/OpenRC/openrc/archive/refs/tags/$version.zip"
depends=""
group="sys.apps,pam"
BUILDDIR="$HOME/distro/build" #Derleme yapılan dizin
DESTDIR="$HOME/distro/rootfs" #Paketin yükleneceği sistem konumu
PACKAGEDIR=$(pwd)
SOURCEDIR="$BUILDDIR/${name}-${version}"

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

setup(){
    mkdir -p $SOURCEDIR/files
    cp $PACKAGEDIR/files/* $SOURCEDIR/files/
    mkdir -p $SOURCEDIR/extras
    cp $PACKAGEDIR/extras/* $SOURCEDIR/extras/
    cd $SOURCEDIR
    meson setup $BUILDDIR \
        --sysconfdir=/etc \
        --prefix=/ \
        --libdir=/lib64 \
        --includedir=/usr/include \
        -Ddefault_library=both \
        -Dzsh-completions=true \
        -Dbash-completions=true \
        -Dpam=true \
        -Dselenium=disabled \
        -Dpkgconfig=true
}

build(){
    #ninja -C $BUILDDIR
    meson compile -C $BUILDDIR
}

package(){
    export DESTDIR=${DESTDIR}
    #DESTDIR=$DESTDIR ninja -C $BUILDDIR install
    DESTDIR="$DESTDIR" meson install --no-rebuild -C $BUILDDIR
}
```

```
# disable all services
rm -f ${DESTDIR}/etc/runlevels/*/*
rm ${DESTDIR}/etc/init.d/functions.sh
ln -s ../../lib/rc/sh/functions.sh ${DESTDIR}/etc/init.d/functions.sh
# install sysconf script
mkdir -p ${DESTDIR}/etc/sysconf.d/
install $SOURCEDIR/files/openrc.sysconf ${DESTDIR}/etc/sysconf.d/openrc

# move /share to /usr/share
```

Paket Derleme

```
mkdir -p ${DESTDIR}/usr ${DESTDIR}/sbin
mv ${DESTDIR}/{,usr}/share
# reboot and poweroff script
install $SOURCEDIR/files/reboot ${DESTDIR}/sbin/reboot
install $SOURCEDIR/files/poweroff ${DESTDIR}/sbin/poweroff
ln -s openrc-shutdown ${DESTDIR}/sbin/shutdown
# install extras
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/modules.init.d ${DESTDIR}/etc/init.d/modules

for level in boot default nonetwork shutdown sysinit ; do
mkdir -p ${DESTDIR}/etc/runlevels/$level
done
touch ${DESTDIR}/etc/fstab
install $SOURCEDIR/files/modules.init.d ${DESTDIR}/etc/init.d/modules
install $SOURCEDIR/files/modules.init.d ${DESTDIR}/etc/runlevels/default/modules

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
#mv ${DESTDIR}/lib ${DESTDIR}/lib64
}
initsetup      # initsetup fonksiyonunu çalıştırır ve kaynak dosyayı indirir
setup          # setup fonksiyonu çalışır ve derleme öncesi kaynak dosyaların ayarlanması sağlanır.
build          # build fonksiyonu çalışır ve kaynak dosyaları derlenir.
package        # package fonksiyonu çalışır, yükleme öncesi ayarlamalar yapılır ve yüklenir.
```

Yukarıdaki kodların sorunsuz çalışabilmesi için ek dosyalara 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 açınız.

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
./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
```

Paket Derleme

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

Paket Derleme

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

```
#!/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"
BUILDDIR="$HOME/distro/build" #Derleme yapılan dizin
DESTDIR="$HOME/distro/rootfs" #Paketin yükleneceği sistem konumu
PACKAGEDIR=$(pwd)
SOURCEDIR="$BUILDDIR/${name}-${version}"

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

setup()
{
    ../${name}-${version}/configure --prefix=/usr \
    --libdir=/lib64/ \
    --with-included-popt \
    --with-included-zlib \
    --disable-xxhash \
    --disable-lz4
}

build()
{
    make
}

package()
{
    make install DESTDIR=$DESTDIR
}

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

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
./build
```

Paket Derleme

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 yapılabilir.

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-${version}.tar.gz"
depends="pam"
makedepend="flex,autoconf,automake"
group="sys.apps"
PACKAGEDIR=$(pwd)
SOURCEDIR="$BUILDDIR/${name}-${version}"

BUILDDIR="$HOME/distro/build" #Derleme yapılan dizin
DESTDIR="$HOME/distro/rootfs" #Paketin yükleneceği sistem konumu

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

setup(){
    mkdir -p $SOURCEDIR/files
    cp $PACKAGEDIR/files/* $SOURCEDIR/files/

    cd $SOURCEDIR
    autoreconf -fvi
    ./configure --prefix=/usr \
        --sysconfdir=/etc \
        --datadir=/usr/share/kbd \
        --enable-optional-progs
}

build(){
    cd ../$name-$version
    pwd
    make KEYCODES_PROGS=yes RESIZECONS_PROGS=yes
}

package(){
    make DESTDIR=${DESTDIR} install
    install -Dm755 $SOURCEDIR/files/loadkeys.initd "$DESTDIR"/etc/init.d/loadkeys
    install -Dm644 $SOURCEDIR/files/loadkeys.conf "$DESTDIR"/etc/conf.d/loadkeys
}

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.
```

Yukarıdaki kodların sorunsuz çalışabilmesi için ek dosyalara 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 açınız.

Paket Derleme

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  
./build
```

busybox

BusyBox, Linux tabanlı sistemlerde yaygın olarak kullanılan, birçok temel Unix aracını bir araya getiren bir yazılım paketidir. "İsviçre Ordusu Bıçağı" benzeri bir işlevsellik sunarak, çeşitli komutları tek bir ikili dosya altında toplar. Bu sayede, sistem kaynaklarını verimli bir şekilde kullanarak, özellikle gömülü sistemlerde ve düşük kaynaklı ortamlarda önemli bir rol oynar.

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, sistem üzerinde etkili bir şekilde işlem yapabilirler. Örneğin, bir dosyayı kopyalamak için aşağıdaki komut kullanılabilir:

```
busybox cp kaynak_dosya hedef_dosya
```


Paket Derleme

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"
BUILDDIR="$HOME/distro/build" #Derleme yapılan dizin
DESTDIR="$HOME/distro/rootfs" #Paketin yükleneceği sistem konumu
PACKAGEDIR=$(pwd)
SOURCEDIR="$BUILDDIR/${name}-${version}"

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

setup()
{
    cp -prfv $PACKAGEDIR/files $SOURCEDIR/

    cd $SOURCEDIR
    make defconfig
    sed -i "s|.*CONFIG_STATIC_LIBGCC .*|CONFIG_STATIC_LIBGCC=y|" .config
    sed -i "s|.*CONFIG_STATIC .*|CONFIG_STATIC=y|" .config
}

build()
{
    make
}

package()
{
    mkdir -p $DESTDIR/bin
    install busybox ${DESTDIR}/bin/busybox
    # install udhcpc script and service
    mkdir -p ${DESTDIR}/usr/share/udhcpc/ ${DESTDIR}/etc/init.d/
    install $SOURCEDIR/files/udhcpc.script ${DESTDIR}/usr/share/udhcpc/default.script
    install $SOURCEDIR/files/udhcpc.openrc ${DESTDIR}/etc/init.d/udhcpc
}

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.
```

Yukarıdaki kodların sorunsuz çalışabilmesi için ek dosyalara 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 açınız.

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
./build
```


kernel

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 oluşturmaya olanak tanır. Özellikle sistem yöneticileri ve geliştiriciler arasında yaygın olarak kullanılan güçlü bir araçtır.

Paket Derleme

Derleme

```
#!/usr/bin/env bash
name="kernel"
version="6.9.9"
description="Linux kernel"
source="https://cdn.kernel.org/pub/linux/kernel/v6.x/linux-$version.tar.xz"
depends=""
builddepend="rsync, bc, cpio, gettext, elfutils, pahole, perl, python, tar, xz-utils"
group="sys.kernel"
BUILDDIR="$HOME/distro/build" #Derleme yapılan dizin
DESTDIR="$HOME/distro/rootfs" #Paketin yükleneceği sistem konumu
PACKAGEDIR=$(pwd)
SOURCEDIR="$BUILDDIR/${name}-${version}"

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

setup(){
    cp -prvf $PACKAGEDIR/files/ $SOURCEDIR/
    cd $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(){
    #-----
    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
```

Paket Derleme

```
install -Dt "$kernelbuilddir/tools/objtool" tools/objtool/objtool
mkdir -p "$kernelbuilddir/{fs/xfstools,mm}"
ln -s "$kernelbuilddir/lib/modules/${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 "$kernelbuilddir/arch/$arch"
cp -v -t "$kernelbuilddir/arch/$arch" -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
# https://bugs.archlinux.org/task/71392
install -Dt "$kernelbuilddir/drivers/iio/common/hid-sensors" -m644 drivers/iio/common/hid-sensors/*.h

find . -name 'Kconfig*' -exec install -Dm644 {} "$kernelbuilddir/{}" \;

# clearing
find -L "$kernelbuilddir" -type l -printf 'Removing %P\n' -delete
find "$kernelbuilddir" -type f -name '*.o' -printf 'Removing %P\n' -delete

if [[ -d "$kernelbuilddir" ]] ; then
    while read -r file; do
        case "$(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 "$kernelbuilddir/vmlinux" ]] ; then
    echo "Stripping vmlinux..."
    strip "$kernelbuilddir/vmlinux"
fi

echo "Adding symlink..."
mkdir -p "$DESTDIR/usr/src"
ln -sr "$kernelbuilddir" "$DESTDIR/usr/src/linux"

#-----
mv -vf System.map $DESTDIR/boot/System.map-$version
find ${DESTDIR}/ -iname "*" -exec unxz {} \;
depmod -b "$DESTDIR" -F $DESTDIR/boot/System.map-$version $version
}
initsetup # initsetup fonksiyonunu çalıştırır ve kaynak dosyayı indirir
setup # setup fonksiyonu çalışır ve derleme öncesi kaynak dosyaların ayarlanması sağlanır.
build # build fonksiyonu çalışır ve kaynak dosyaları derlenir.
package # package fonksiyonu çalışır, yükleme öncesi ayarlamalar yapılır ve yüklenir.
```

Yukarıdaki kodların sorunsuz çalışabilmesi için ek dosyalara 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 açınız.

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

Paket Derleme

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

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

kernel-headers

Kernel-headers paketi, Linux çekirdeği ile kullanıcı alanı uygulamaları arasında bir köprü işlevi gören başlık dosyalarını barındırır. Bu dosyalar, sistem programcılarının ve geliştiricilerin, çekirdek işlevlerine erişim sağlamak için gerekli olan API'leri ve veri yapılarını tanımlar. Özellikle, sürücü geliştirme ve sistem yazılımları için kritik öneme sahiptir.

Paket Derleme

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-$version.tar.xz"
depends="kernel"
builddepend="rsync, bc, cpio, gettext, elfutils, pahole, perl, python, tar, xz-utils"
group="sys.kernel"
BUILDDIR="$HOME/distro/build" #Derleme yapılan dizin
DESTDIR="$HOME/distro/rootfs" #Paketin yükleneceği sistem konumu
PACKAGEDIR=$(pwd)
SOURCEDIR="$BUILDDIR/${name}-${version}"

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

setup(){
    cp -prvf $PACKAGEDIR/files/ $SOURCEDIR/
    cd $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(){
    #-----
    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
```


Paket Derleme

```
install -Dt "$kernelbuilddir/tools/objtool" tools/objtool/objtool
mkdir -p "$kernelbuilddir/{fs/xfstools,mm}"
ln -s "$kernelbuilddir/lib/modules/${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 "$kernelbuilddir/arch/$arch"
cp -v -t "$kernelbuilddir/arch/$arch" -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
# https://bugs.archlinux.org/task/71392
install -Dt "$kernelbuilddir/drivers/iio/common/hid-sensors" -m644 drivers/iio/common/hid-sensors/*.h

find . -name 'Kconfig*' -exec install -Dm644 {} "$kernelbuilddir/{}" \;

# clearing
find -L "$kernelbuilddir" -type l -printf 'Removing %P\n' -delete
find "$kernelbuilddir" -type f -name '*.o' -printf 'Removing %P\n' -delete

if [[ -d "$kernelbuilddir" ]] ; then
    while read -r file; do
        case "$(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 "$kernelbuilddir/vmlinux" ]] ; then
    echo "Stripping vmlinux..."
    strip "$kernelbuilddir/vmlinux"
fi

echo "Adding symlink..."
mkdir -p "$DESTDIR/usr/src"
ln -sr "$kernelbuilddir" "$DESTDIR/usr/src/linux"

#-----
mv -vf System.map $DESTDIR/boot/System.map-$version
find $DESTDIR/ -iname "*" -exec unxz {} \;
depmod -b "$DESTDIR" -F $DESTDIR/boot/System.map-$version $version
}
initsetup # initsetup fonksiyonunu çalıştırır ve kaynak dosyayı indirir
setup # setup fonksiyonu çalışır ve derleme öncesi kaynak dosyaların ayarlanması sağlanır.
build # build fonksiyonu çalışır ve kaynak dosyaları derlenir.
package # package fonksiyonu çalışır, yükleme öncesi ayarlamalar yapılır ve yüklenir.
```

Yukarıdaki kodların sorunsuz çalışabilmesi için ek dosyalara ihtiyaç vardır. Bu ek dosyaları indirmek için [tıklayınız](#).

tar dosyasını indirdikten sonra istediğiniz bir konumda **kernel-headers** adında bir dizin oluşturun ve tar dosyasını oluşturulan dizin içinde açınız.

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

Paket Derleme

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

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

Paket Derleme

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

```
#!/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"
BUILDDIR="$HOME/distro/build" #Derleme yapılan dizin
DESTDIR="$HOME/distro/rootfs" #Paketin yükleneceği sistem konumu
PACKAGEDIR=$(pwd)
SOURCEDIR="$BUILDDIR/${name}-${version}"

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

setup()
{
    echo "test"
    cd $SOURCEDIR
}

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
}

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(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
./build
```

Paket Derleme

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"
BUILDDIR="$HOME/distro/build" #Derleme yapılan dizin
DESTDIR="$HOME/distro/rootfs" #Paketin yükleneceği sistem konumu
PACKAGEDIR=$(pwd)
SOURCEDIR="$BUILDDIR/${name}-${version}"

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

setup()
{
    cd $SOURCEDIR
}

build()
{
    make
}

package()
{
    make install DESTDIR=$DESTDIR
}

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

Paket 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
./build
```

Paket Derleme

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

```
#!/usr/bin/env bash
version="3.6"
name="parted"
depends="glibc"
description="disks tools"
source="https://ftp.gnu.org/gnu/parted/parted-${version}.tar.xz"
groups="sys.block"
BUILDDIR="$HOME/distro/build" #Derleme yapılan dizin
DESTDIR="$HOME/distro/rootfs" #Paketin yükleneceği sistem konumu
PACKAGEDIR=$(pwd)
SOURCEDIR="$BUILDDIR/${name}-${version}"

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

setup()
{
    ../${name}-${version}/configure --prefix=/usr \
    --libdir=/usr/lib64/ \
    --sbindir=/usr/bin \
    --disable-rpath \
    --disable-device-mapper
}

build()
{
    make
}

package()
{
    make install DESTDIR=$DESTDIR
}

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

Paket 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
./build
```

Paket Derleme

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 oluşturmaya olanak tanır. Özellikle sistem yöneticileri ve geliştiriciler arasında yaygın olarak kullanılan güçlü bir araçtır.

Derleme

```
#!/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
BUILDDIR="$HOME/distro/build" #Derleme yapılan dizin
DESTDIR="$HOME/distro/rootfs" #Paketin yükleneceği sistem konumu
PACKAGEDIR=$(pwd)
SOURCEDIR="$BUILDDIR/${name}-${version}"

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

setup(){
    touch libkmod/docs/gtk-doc.make
    $SOURCEDIR/configure --prefix=/usr \
        --libdir=/usr/lib64/ \
        --bindir=/bin \
        --with-rootlibdir=/lib \
        --with-zlib \
        --with-openssl
}

build(){
    make
}

package(){
    make install DESTDIR=$DESTDIR
    echo "Dizin : "
    pwd
    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
}

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

Paket Derleme

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  
./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.

Paket Derleme

nano

Nano, terminal tabanlı bir metin düzenleyici olup, GNU projesinin bir parçasıdır. Kullanıcıların metin dosyalarını kolayca oluşturmalarına, 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"
BUILDDIR="$HOME/distro/build" #Derleme yapılan dizin
DESTDIR="$HOME/distro/rootfs" #Paketin yükleneceği sistem konumu
PACKAGEDIR=$(pwd)
SOURCEDIR="$BUILDDIR/${name}-${version}"

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

setup()
{
    cd $SOURCEDIR
    ./configure --prefix=/usr
}

build()
{
    make
}

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

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(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
./build
```


grub

GRUB (GRand Unified Bootloader), çoklu işletim sistemlerini destekleyen ve kullanıcıların sistemlerini başlatmalarını sağlayan bir önyükleyici yazılımdır. Linux tabanlı sistemlerde yaygın olarak kullanılan GRUB, sistem açılışında gerekli olan çekirdek dosyalarını yükleyerek işletim sisteminin çalışmasını başlatır. GRUB, kullanıcıların farklı işletim sistemleri arasında seçim yapmalarına olanak tanırken, aynı zamanda gelişmiş yapılandırma seçenekleri sunar.

GRUB'un temel özellikleri arasında, çoklu çekirdek desteği, ağ üzerinden önyükleme yapabilme yeteneği ve kullanıcı dostu bir arayüz bulunmaktadır. Örneğin, GRUB yapılandırma dosyası genellikle `/boot/grub/grub.cfg` konumunda bulunur ve burada önyükleme seçenekleri tanımlanır.

Paket Derleme

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
}

BUILDDIR="$HOME/distro/build" #Derleme yapılan dizin
DESTDIR="$HOME/distro/rootfs" #Paketin yükleneceği sistem konumu
PACKAGEDIR=$(pwd)
SOURCEDIR="$BUILDDIR/${name}-${version}"

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

setup(){
    cd /tmp/bps/build
    echo depends bli part_gpt > $SOURCEDIR/grub-core/extra_deps.lst
    for tgt in ${uses[@]} ; do
        # if ! use $tgt ; then
        #     continue
        # fi
        cp -prfv $name-$version $tgt
    done
}
```

done

```
for tgt in ${uses[@]} ; do
    #if ! use $tgt ; then
    #    continue
    # fi
    cd $tgt
    autoreconf -fvi
    ./configure --prefix=/usr \
        --sysconfdir=/etc \
```

Paket Derleme

```
--libdir=/usr/lib64/ \
--disable-nls \
--disable-werror \
--disable-grub-themes \
$(get_grub_opt $tgt)
cd ..
done
}

build(){
  for tgt in ${uses[@]} ; do
    #if ! use $tgt ; then
    #   continue
    #fi
    make $jobs -C $tgt
  done
}

package(){
  for tgt in ${uses[@]} ; do
    # if ! use $tgt ; then
    #   continue
    #fi
    make $jobs -C $tgt install DESTDIR=$DESTDIR
  done
  # default grub config
  mkdir -p $DESTDIR/etc/default $DESTDIR/usr/bin/
  {
    echo 'GRUB_DISTRIBUTOR=""'
    echo 'GRUB_TERMINAL_OUTPUT=console'
    echo 'GRUB_CMDLINE_LINUX_DEFAULT="quiet"'
    echo 'GRUB_CMDLINE_LINUX=""'
    echo 'GRUB_DEFAULT=0'
    echo 'GRUB_TIMEOUT=5'
    echo 'GRUB_DISABLE_SUBMENU=y'
    echo 'GRUB_DISABLE_OS_PROBER=true'
    echo 'GRUB_DISABLE_RECOVERY=true'
  } > $DESTDIR/etc/default/grub
  echo "#!/bin/sh" > $DESTDIR/usr/bin/update-grub
  echo "grub-mkconfig -o /boot/grub/grub.cfg" >> $DESTDIR/usr/bin/update-grub
  chmod 755 $DESTDIR/usr/bin/update-grub
}

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

Paket adında(grub) 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
./build
```

Paket Derleme

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

```
#!/usr/bin/env bash
name="efibootmgr"
version="18"
description="Linux user-space application to modify the Intel Extensible Firmware Interface (EFI) Boot Manager."
source="https://github.com/rhboot/efibootmgr/archive/refs/tags/${version}.tar.gz"
depends="efivar,popt"
builddepends=""
group="sys.boot"
BUILDDIR="$HOME/distro/build" #Derleme yapılan dizin
DESTDIR="$HOME/distro/rootfs" #Paketin yükleneceği sistem konumu
PACKAGEDIR=$(pwd)
SOURCEDIR="$BUILDDIR/${name}-${version}"

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

setup(){
    echo ""
}

build(){
    cd $SOURCEDIR
    make sbindir=/usr/bin EFIDIR=/boot/efi PCDIR=/usr/lib64/pkgconfig
}

package(){
    EFIDIR="/boot/efi" sbindir=/usr/bin make DESTDIR="$DESTDIR" install
}

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(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
./build
```

Paket Derleme

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

```
#!/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/$version.tar.gz"
depends=""
builddepend=""
group="sys.libs"
BUILDDIR="$HOME/distro/build" #Derleme yapılan dizin
DESTDIR="$HOME/distro/rootfs" #Paketin yükleneceği sistem konumu
PACKAGEDIR=$(pwd)
SOURCEDIR="$BUILDDIR/${name}-${version}"

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

setup(){
    export ERRORS=''
    export PATH=$PATH:$HOME
    # fake mandoc for ignore extra dependency
    echo "exit 0" > $HOME/mandoc
    chmod +x $HOME/mandoc
}

build(){
    cd $SOURCEDIR
    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
}

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(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.

Paket Derleme

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

Paket Derleme

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"
BUILDDIR="$HOME/distro/build" #Derleme yapılan dizin
DESTDIR="$HOME/distro/rootfs" #Paketin yükleneceği sistem konumu
PACKAGEDIR=$(pwd)
SOURCEDIR="$BUILDDIR/${name}-${version}"

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

setup()
{
    ../${name}-${version}/configure --prefix=/usr \
    --libdir=/lib64/ \
    --with-ncursesw
    # change default color blue to red
    cd $SOURCEDIR
    # sed -i "s/COLOR_BLUE/COLOR_RED/g" dlg_colors.h
    # sed -i "s/COLOR_CYAN/COLOR_MAGENTA/g" dlg_colors.h
}

build()
{
    cd $BUILDDIR
    make
}

package()
{
    make install DESTDIR=$DESTDIR
}

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

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
./build
```

Paket Derleme

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-${version}.tar.xz")
group="net.libs"
depends="openssl,zlib"
BUILDDIR="$HOME/distro/build" #Derleme yapılan dizin
DESTDIR="$HOME/distro/rootfs" #Paketin yükleneceği sistem konumu
PACKAGEDIR=$(pwd)
SOURCEDIR="$BUILDDIR/${name}-${version}"

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

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/
}

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(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.

Paket Derleme

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

Paket Derleme

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 iletişimin güvenliğini artırır.

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

Derleme

```
#!/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"
BUILDDIR="$HOME/distro/build" #Derleme yapılan dizin
DESTDIR="$HOME/distro/rootfs" #Paketin yükleneceği sistem konumu
PACKAGEDIR=$(pwd)
SOURCEDIR="$BUILDDIR/${name}-${version}"

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

setup(){
    cp -prfv $PACKAGEDIR/files/sshd.initd $BUILDDIR/
    cp -prfv $PACKAGEDIR/files/sshd.conf.d $BUILDDIR/
    cd $BUILDDIR
    ../${name}-${version}/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(){
    cd $BUILDDIR
    make install DESTDIR=$DESTDIR
    mkdir -p "$DESTDIR"/etc/{passwd,group,sysconf,init,conf}.d
    install -m755 -D sshd.initd "$DESTDIR"/etc/init.d/sshd
    install -m755 -D sshd.conf.d "$DESTDIR"/etc/conf.d/sshd
}

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

Paket Derleme

Yukarıdaki kodların sorunsuz çalışabilmesi için ek dosyalara 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 açınız.

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  
./build
```

Paketler derlendikten sonra files dizini içindeki postinstall scriptinin çalıştırılması gerekmektedir. Bu dosya "\$HOME/distro/rootfs" konumunda chroot ile çalıştırılmalıdır.

Paket Sistemi

Paket Sitemi

Paket yönetim sistemleri bir dağıtımda bulunan en temel parçadır. Sistem üzerine paket kurma ve kaldırma güncelleme yapma gibi işlemlerden sorumludur. Başlıca 2 tip paket sistemi vardır:

- Binary (ikili) paket sistemi
- Source (kaynak) paket sistemi

Bir paket sistemi hem binary hem source paket sistemi özelliklerine sahip olabilir. Bununla birlikte son kullanıcı dağıtımlarında genellikle binary paket sistemleri tercih edilir.

Binary Paket Sistemi

Bu tip paket sistemlerinde önceden derlenmiş olan paketler hazır şekilde indirilir ve açılarak sistem ile birleştirilir. Binary paket sistemlerinde paketler önceden derleme talimatları ile oluşturulmalıdır.

Binary paket sistemine örnek olarak **apt**, **dnf**, **pacman** örnek verilebilir.

Source Paket Sistemi

Bu tip paket sistemlerinde derleme talimatları kurulum yapılacak bilgisayar üzerinde kullanarak paketler kurulum yapılacak bilgisayarda oluşturulur ve kurulur.

Source paket sistemine örnek olarak **portage** örnek verilebilir.

Paket sisteminin temel yapısı

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

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

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

Bazı dağıtımların kullandığı paket sistemleri şunlardır.

- apt: Debian dağıtımının kullandığı paket sistemi.
- emerge :Gentoo dağıtımının kullandığı paket sistemi.
- ymp : Turkman Linux dağıtımının kullandığı paket sistemi.

bps Paket Sistemi

Bu dokümanda hazırlanan dağıtımın paket sistemi için ise bps(basit/basic/base paket sistemi) olarak ifade edeceğimiz paket sistemi adını kullandık. Bps paket sistemindeki beş temel işlemin nasıl yapılacağı ayrı başlıklar altında anlatılacaktır. Paket sistemi delemeli bir dil yerine bash script ile yapılacaktır. Bu dokümanı takip eden orta seviye bilgiye sahip olan linux kullanıcısı yapılan işlemleri anlaması amaçlandı.

Paket Oluşturma

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

Dokümanda readline paketi nasıl derleneceği aşağıdaki script yapılıyor.

```
# kaynak kod indirme ve derleme için hazırlama
version="8.1"
name="readline"
mkdir -p $HOME/distro
cd $HOME/distro
rm -rf ${name}-${version}
rm -rf build-${name}-${version}
wget https://ftp.gnu.org/pub/gnu/readline/${name}-${version}.tar.gz
tar -xvf ${name}-${version}.tar.gz
mkdir build-${name}-${version}
cd build-${name}-${version}
../${name}-${version}/configure --prefix=/ --enable-shared --enable-multibyte

# derleme
make

# derlenen paketin yüklenmesi ve ayarlamaların yapılması
make install DESTDIR=$HOME/rootfs
```

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 **bpspaketle** ve **bpsbuild** adlı script dosyaları olarak düzenleyeceğiz. Genel yapısı aşağıdaki gibi olacaktır.

bpsbuild Dosyası

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

bpspaketle Dosyası

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

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

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

Paket Sistemi

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

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

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

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

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

bpsbuild Dosyamızın Son Hali

```
#!/usr/bin/env bash
version="8.1"
name="readline"
depends="glibc"
description="readline kütüphanesi"
source="https://ftp.gnu.org/pub/gnu/readline/${name}-${version}.tar.gz"
groups="sys.apps"
setup()
{
    ../${name}-${version}/configure --prefix=/ --enable-shared --enable-multibyte
}
build()
{
    make
}
package()
{
    make install DESTDIR=$DESTDIR
}
```

bpspaketle Dosyamızın Son Hali

```
#!/usr/bin/env bash
set -e
paket=$1
dizin=$(pwd)
if [ ! -d ${paket} ]; then echo "Bir paket değil!"; exit; fi
if [ ! -f "${paket}/bpsbuild" ]; then echo "Paket dosyası bulunamadı!"; exit; fi
echo "Paket : $paket"
source ${paket}/bpsbuild
DESTDIR=/tmp/bps/build/rootfs-${name}-${version}
SOURCEDIR=/tmp/bps/build/${name}-${version}
BUILDDIR=/tmp/bps/build/build-${name}-${version}

# paketin indirilmesi ve /tmp/bps/build konumunda derlenmesi için gerekli izinler hazırlanır.
initsetup()
{
    mkdir -p /tmp/bps
    mkdir -p /tmp/bps/build
    cd /tmp/bps/build
    rm -rf ./.*
    rm -rf build-${name}-${version}*
    rm -rf ${name}-${version}*
    rm -rf rootfs-${name}-${version}*

    if [ -n "${source}" ]
    then
        wget ${source}
        downloadfile=$(ls|head -1)
        filetype=$(file -b --extension $downloadfile|cut -d'/' -f1)
        echo "*****dosya sıkıştırma türü*****:${filetype}"
        if [ ${filetype} == "bz2" ]; then tar -xvf ${downloadfile}; fi
        if [ ${filetype} == "tar" ]; then tar -xvf ${downloadfile}; fi
        if [ ${filetype} == "xz" ]; then tar -xvf ${downloadfile}; fi
        if [ "${filetype}" == "gz" ]; then echo "*****dosya gz ile sıkıştırılmış**"; tar -xvf ${downloadfile}; fi
        if [ "${filetype}" == "???" ]; then echo "*****dosya zip ile sıkıştırılmış*****"; unzip ${downloadfile}; fi
        #*****
        director=$(find ./.* -maxdepth 0 -type d)
        if [ "${director}" != "./${name}-${version}" ]; then mv $director ${name}-${version}; fi
    fi
    mkdir -p build-${name}-${version}
    mkdir -p rootfs-${name}-${version}
    cp ${dizin}/${paket}/bpsbuild /tmp/bps/build
    cd build-${name}-${version}
}

#paketlenecek dosyaların listesini tutan file.index dosyası oluşturulur
packageindex()
{
    rm -rf file.index
    cd /tmp/bps/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;
# kurulacak data rootfs.tar.xz, file.index ve bpsbuild dosyaları tek bir dosya olarak tar.gz dosyası olarak hazırlanıyor.
# tar.gz dosyası olarak hazırlanan dosya bps ismiyle değiştirilip paketimiz hazırlanır.

packagecompress()
{
    cd /tmp/bps/build/rootfs-${name}-${version}
    tar -cf ../rootfs.tar ./.*
    cd /tmp/bps/build/
    xz -9 rootfs.tar
    tar -cvzf paket-${name}-${version}.tar.gz rootfs.tar.xz file.index bpsbuild
    cp paket-${name}-${version}.tar.gz ${dizin}/${paket}/${name}-${version}.bps
}

# fonksiyonlar aşağıdaki sırayla çalışacaktır.
echo "***** initsetup *****"; initsetup #bu dosya içindeki fonksiyon
echo "***** setup *****"; setup #bpsbuild dosyasından gelen fonksiyon
echo "***** build *****"; build #bpsbuild dosyasından gelen fonksiyon
echo "***** package *****"; package #bpsbuild dosyasından gelen fonksiyon
echo "***** packageindex *****"; packageindex #bu dosya içindeki fonksiyon
echo "***** packagecompress *****"; packagecompress #bu dosya içindeki fonksiyon
```

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

Paket Yapma

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

```
mkdir readline
cd readline
#readline için hazırlanan bpsbuild dosyası bu konuma oluşturulur ve içeriği readline için oluşturduğumuz bpsbuild içeriği olarak ayarlanır.
cd ..
./bpspaketle readline # bpspaketle dosyamızın bu konumda olduğu varsayılmıştır ve parametre olarak readline dizini verilmiştir.
```

Komut çalışınca readline/readline-8.1.bps dosyası oluşacaktır. 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, paket yönetim sistemlerinde kurulacak olan paketleri içeren bir veri topluluğudur. Kaynak depo ve ikili depo olarak ikiye ayrılır. Depo içerisinde hiyerarşik olarak paketler yer alır. Index ise depoda yer alan paketlerin isimleri sürüm numaraları gibi bilgiler ile adreslerini tutan kayıttır. Paket yönetim sistemi index içerisinden gelen veriye göre gerekli paketi indirir ve kurar. Depo indexi aşağıdaki gibi olabilir:

```
Package: hello
Version: 1.0
Dependencies: test, foo, bar
Path: h/hello/hello_1.0_x86_64.zip

Package: test
Version: 1.1
Path: t/test/test_1.1_aarch64.zip

...
```

Yukarıdaki örnekte paket adı bilgisi sürüm bilgisi ve bağımlılıklar 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 **.INFO** dosyası taşıyın. Aşağıdaki gibi depo indexi alabiliriz.

```
function index {
    > index.txt
    for i in $@ ; do
        unzip -p $i .INFO >> index.txt
        echo "Path: $i" >> index.txt
    done
}
index t/test/test_1.0_x86_64.zip h/hello/hello_1.1_aarch64.zip ...
```

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

bps Paket Liste Indexi Güncelleme

Dağıtımlarda uygulamalar paketler halinde hazırlanır. Bu paketleri isimleri, versiyonları ve bağımlılık gibi temel bilgileri barındıran liste halinde tutan bir dosya oluşturulur. Bu dosyaya **index.lst** isim verebiliriz. Bu dokümanda bu listeyi tutan **index.lst** dosyası kullanılmıştır. Paket sisteminde güncelleme aslında **index.lst** dosyanın en güncellen halinin sisteme yüklenmesi olayıdır.

bps paketleme sisteminde **bpsupdate** scripti hazırlanmıştır. Bu script **index.lst** dosyasının paketlerimizin en güncel halini sistemimize yükleyecektir. Bu dağıtımda paketlerimizi github.com üzerinde oluşturulan bir repository üzerinden çekilmektedir. Paket listemiz ise yapılan her yeni paketi yükleme sırasında güncellenmektedir.

Paket güncelleme için iki script kullanılmaktadır. Bunlar;

index.lst Dosyasını Oluşturma

```
#index alma scripti
#!/bin/sh
set -ex
>index.lst
find ./ -type f -name *bps |
    while IFS= read file_name; do
        tar -xf ${file_name} bpsbuild
        version=$(cat bpsbuild|grep version=)
        name=$(cat bpsbuild|grep name=)
        depends=$(cat bpsbuild|grep depends=)
        echo "$name:$version:$depends">>index.lst
    done
rm -rf bpsbuild
mkdir /output -p
cp -rf index.lst /output
```

Bu script bps paket dosyalarımızın olduğu dizinde tüm paketleri açarak içerisinden **bpsbuild** dosyalarını çıkartarak pakete 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 : karakteri kullanılmıştır.

```
name="glibc":version="2.38":depends=""
name="gmp":version="6.3.0":depends="glibc,readline,ncurses"
name="grub":version="2.06":depends="glibc,readline,ncurses"
name="kmod":version="31":depends="glibc,zlib"
```

index.lst Dosyasını Güncelleme

bpsupdate dosya içeriği

```
#!/bin/sh
curl -O /tmp/index.lst https://basitsadigitim.github.io/binary-package/index.lst
```

index.lst dosyamızı github üzerinden indiren scriptimiz tek bir satırdan oluşmaktadır. Bu komut <https://basitsadigitim.github.io/binary-package/index.lst> adresindeki dosyayı index.lst dosyasını /tmp/index.lst konumuna indirecektir.

Paket Kurma

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.

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

```
cd /onbellek/dizini
unzip /dosya/yolu/paket.zip
cp -rpf ./* /
cp .LIST /paket/veri/yolu/paket.LIST
```

Bu örnekte ilk satırda geçici dizine gittik ve paketi oraya açtık. Daha sonra paket içeriğini kök dizine kopyaladık. Daha sonra paket dosya listesini verilerin tutulduğu yere kopyaladık. Bu işlemden sonra paket kurulmuş oldu.

bps Paket Kurma Scripti Tasarlama

Hazırlanan dağıtımda paketlerin kurulması için sırasıyla aşağıdaki işlem adımları yapılmalıdır.

1. Paketin indirilmesi
2. İndirilen paketin /tmp/bps/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.

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

Paket Sistemi

bpskur Scripti

```
#!/bin/sh
#set -e
paket=$1
paketname="name=\"${paket}\""
ROOTFS=$2
#echo "$paket"
indexpaket=$(cat /tmp/index.lst|grep $paketname)
name=""
version=""
depends=""
if [ -n "${indexpaket}" ]
then
    namex=$(echo $indexpaket|cut -d":" -f1)
    versionx=$(echo $indexpaket|cut -d":" -f2)
    dependsx=$(echo $indexpaket|cut -d":" -f3)
    name=${namex:6:-1}
    version=${versionx:9:-1}
    depends=${dependsx:9:-1}
else
    echo "*****Paket Bulunamadı*****"; exit
fi

# paketi indirme
mkdir -p /tmp/bps
mkdir -p /tmp/bps/kur
rm -rf /tmp/bps/kur/*
./indirgentoo /tmp/bps/kur/${name}-${version}.tar.gz https://github.com/bayramkarahan/distro-binary-package/raw/master/${name}/${name}-${version}.bps
mkdir -p /var/lib/bps
cd /tmp/bps/kur/

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

# paketi kurma
cp -prfv rootfs/* $ROOTFS/

#name version depends /var/bps/index.lst eklenmesi
echo "name=\"${name}\"version=\"${version}\"depends=\"${depends}\"">>var/bps/index.lst
#paket içinde gelen paket dosyalarının dosya ve dizin yapısını tutan file index dosyasının /var/bps/ konumuna kopyalanması
cp file.lst /var/bps/${name}-${version}.lst
```

bpskur 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.

```
./bpskur readline /
```

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 paket listesi dosyamızı sildik. Bu işlem sonunda paket silinmiş oldu.

bps Paket Kaldırma Scripti Tasarlama

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

bpskaldir scripti

```
#!/bin/sh
#set -e
paket=$1
paketname="name=\"${paket}\""
indexpaket=$(cat /var/bps/index.lst|grep $paketname)
name=""
version=""
depends=""
if [ -n "${indexpaket}" ]; then
    namex=$(echo $indexpaket|cut -d":" -f1)
    versionx=$(echo $indexpaket|cut -d":" -f2)
    dependsx=$(echo $indexpaket|cut -d":" -f3)
    name=${namex:6:-1}
    version=${versionx:9:-1}
    depends=${dependsx:9:-1}
else
    echo "*****Paket Bulunamadı*****"; exit
fi
# Bağımlılıkları başka paketler kullanıyor mu kontrol edilir
echo "bağımlılık kontrolü yapılacak"

# Paketin file.lst dosyası içerisindeki dosyalar, dizinler kaldırılır.
cat /var/bps/${paket}-${version}.lst | while read dosya ; do if [[ -f "$dosya" ]] ; then rm -f "$dosya"; fi done
cat /var/bps/${paket}-${version}.lst | while read dizin ; do if [[ -d "$dizin" ]] ; then rmdir "$dizin" || true; fi done

# /var/bps/paket-version.lst dosyasından paket bilgisi kaldırılır.
rm -f /var/bps/${paket}-${version}.lst
# /var/bps/index.lst dosyasından ilgili paket satırı kaldırılır.
sed '/^name=\"${paket}\"/d' /var/bps/index.lst
```

Paket Sistemi

Bağımlılıkları başka paketler kullanıyor mu kontrol edilir. Script içinde bu işlem yapılmamıştır. Daha sonra güncellenecektir. Bu örnekte paket listesini satır satır okuduk. Önce dosya olanları sildik. Daha sonra tekrar okuyup boş kalan izinleri sildik. Son olarak paket listesi dosyamızı sildik. Bu işlem sonunda paket silinmiş oldu.

bpskaldir Kullanma

bpskaldir scripti aşağıdaki gibi kullanılır.

```
./bpskaldir readline
```

iso Hazırlama

İso Hazırlama

initrd hazırlama aşamaları **initrd** konu başlığında detaylıca anlatıldı. Sistem hazırlanırken küçük farklılıklar olsada **initrd** hazırlamaya benzer aşamalar yapılacaktır. Sistemimin yani oluşacak **iso** dosyasının yapısı aşağıdaki gibi olacaktır. Aşağıda sadece **filesystem.squashfs** dosyasının hazırlanması kaldı.

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

filesystem.squashfs Hazırlama

filesystem.squashfs dosyası **/initrd.img** dosyasına benzer yapıda hazırlanacak. En büyük farklılık **init** çalışabilir dosya içeriğinde yapılmalı. Yapı **/initrd.img** dizin yapısı gibi hazırlandıktan sonra **filesystem.squashfs** oluşturulmalı ve **\$HOME/distro/iso/live/filesystem.squashfs** konuma kopyalanmalıdır. Aşağıdaki komutlarla **filesystem.squashfs** hazırlanıyor ve **\$HOME/distro/iso/live/** konumuna taşınıyor.

```
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

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

Artık sistemi açabilen ve tty açıp bize sunan bir yapı oluşturduk. Çalıştırmak için qemu kullanılabilir.

qemu-system-x86_64 -cdrom distro.iso -m 1G komutuyla çalıştırıp test edebiliriz.

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.

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 modülleri **modprobe** komutuyla yüklenmeli.

Disk Hazırlanmalı(legacy)

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

0. cfdisk komutuyla disk bölümlendirilmeli. .. code-block:: shell

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

Şimdi de disk bölümümüzü bağlayalım.

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

Ardından dosyaları kopyalayalım.

```
# -p dosya izinlerini korur
# -r alt dizinlerle beraber kopyalar
# -f soru sormayı kapatır
# -v detaylı çıktıları gösterir
$ cp -prfv /source/* /target
```

Daha sonra diski senkronize edelim.

Sistem Kurulumu

```
$ sync
```

Bootloader kurulumu

grub kurulumu yapmak için grub paketini kurulu olduğundan emin olun.

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

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

Grub Kuralım

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

Grub yapılandırması

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

```
$ blkid | grep /dev/sda1
/dev/sda1: UUID="..." BLOCK_SIZE="4096" TYPE="ext4" PARTUUID="..."
```

Şimdi aşağıdaki gibi bir yapılandırma dosyası yazalım ve /boot/grub/grub.cfg dosyasına kaydedelim. Burada uuid değerini ve çekirdek sürümünü düzenleyin.

```
linux /boot/vmlinuz-<çekirdek-sürümü> root=UUID=<uuid-değeri> rw quiet
initrd /boot/initrd.img-<çekirdek-sürümü>
boot
```

Ayrıca otomatik yapılandırma da oluşturabiliriz.

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

Sistem Kurulumu

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	

Not: Disk bölümü konumu yerine **UUID=<uuid-değeri>** şeklinde yazmanızı öneririm. Bölüm adları değişebilirken uuid değerleri değişmez.

İki Bölüm Kurulum

Bu bölümde **Ext4** dosya sistemine grub kullanarak kurulum anlatılacaktır. Anlatım boyunca **/dev/sda** diski üzerinden örneklemeye yapılmıştır. Siz kendi diskinize göre düzenleyebilirsiniz. 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 modülleri **modprobe** komutuyla yüklenmeli.

Disk Hazırlanmalı

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

0. cfdisk komutuyla disk bölümlendirilmeli. .. code-block:: shell

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

Şimdi de disk bölümümüzü bağlayalım.

Sistem Kurulumu

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

Ardından dosyaları kopyalayalım.

```
# -p dosya izinlerini korur
# -r alt dizinlerle beraber kopyalar
# -f soru sormayı kapatır
# -v detaylı çıktıları gösterir
$ cp -prfv /source/* /target
```

Daha sonra diski senkronize edelim.

```
$ sync
```

Bootloader kurulumu

grub kurulumu yapmak için grub paketinin kurulu olduğundan emin olun.

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

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

Grub Kuralım

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

Grub yapılandırması

1. /boot bölümünde initrd.img-<çekirdek-sürümü> dosyamızın olduğundan emin olalım.
2. /boot bölümünde vmlinuz-<çekirdek-sürümü> 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.

Sistem Kurulumu

```
$ blkid | grep /dev/sda2  
/dev/sda2: UUID="..." BLOCK_SIZE="4096" TYPE="ext4" PARTUUID="..."
```

Şimdi aşağıdaki gibi bir yapılandırma dosyası yazalım ve /boot/grub/grub.cfg dosyasına kaydedelim. Burada uuid değerini ve çekirdek sürümünü düzenleyin.

```
linux /vmlinuz-<çekirdek-sürümü>          root=UUID=<uuid-değeri> rw quiet  
initrd /initrd.img-<çekirdek-sürümü>  
boot
```

Ayrıca otomatik yapılandırma da oluşturabiliriz.

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

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

Not: Disk bölümü konumu yerine **UUID=<uuid-değeri>** şeklinde yazmanızı öneririm. Bölüm adları değişebilirken uuid değerleri değişmez.

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 modülleri **modprobe** komutuyla yüklenmeli.

Disk Hazırlanmalı(legacy)

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

0. cfdisk komutuyla disk bölümlendirilmeli. .. code-block:: shell

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

Şimdi de disk bölümümüzü bağlayalım.

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

Ardından dosyaları kopyalayalım.

```
# -p dosya izinlerini korur
# -r alt dizinlerle beraber kopyalar
# -f soru sormayı kapatır
# -v detaylı çıktıları gösterir
$ cp -prfv /source/* /target
```

Daha sonra diski senkronize edelim.

Sistem Kurulumu

```
$ sync
```

Bootloader kurulumu

grub kurulumu yapmak için grub paketini kurulu olduğundan emin olun.

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

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

Grub Kuralım

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

Grub yapılandırması

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

```
$ blkid | grep /dev/sda1
/dev/sda1: UUID="..." BLOCK_SIZE="4096" TYPE="ext4" PARTUUID="..."
```

Şimdi aşağıdaki gibi bir yapılandırma dosyası yazalım ve /boot/grub/grub.cfg dosyasına kaydedelim. Burada uuid değerini ve çekirdek sürümünü düzenleyin.

```
linux /boot/vmlinuz-<çekirdek-sürümü> root=UUID=<uuid-değeri> rw quiet
initrd /boot/initrd.img-<çekirdek-sürümü>
boot
```

Ayrıca otomatik yapılandırma da oluşturabiliriz.

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


Sistem Kurulumu

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	

Not: Disk bölümü konumu yerine **UUID=<uuid-değeri>** şeklinde yazmanızı öneririm. Bölüm adları değişebilirken uuid değerleri değişmez.

Uefi Sistem Kurulumu

Bu bölümde **Ext4** dosya sistemine grub kullanarak kurulum anlatılacaktır. Anlatım boyunca **/dev/sda** diski üzerinden örneklemeye yapılmıştır. Siz kendi diskinize göre düzenleyebilirsiniz. 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 modülleri **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.

Sistem Kurulumu

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

Şimdi de disk bölümümüzü bağlayalım.

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

Ardından dosyaları kopyalayalım.

```
# -p dosya izinlerini korur
# -r alt dizinlerle beraber kopyalar
# -f soru sormayı kapatır
# -v detaylı çıktıları gösterir
$ cp -prfv /source/* /target
```

Daha sonra diski senkronize edelim.

```
$ sync
```

Bootloader kurulumu

grub kurulumu yapmak için grub paketini kurulu olduğundan emin olun.

```
$ mkdir -p /target/dev
$ mkdir -p /target/sys
$ mkdir -p /target/proc
$ mkdir -p /target/run
$ mkdir -p /target/tmp
$ mount --bind /dev /target/dev
$ mount --bind /sys /target/sys
$ mount --bind /proc /target/proc
$ mount --bind /run /target/run
$ mount --bind /tmp /target/tmp
#efi alan bağlanıyor. Eğer uefi aktif edilmişse kernel **/sys/firmware/efi** tarafından budizinler ve dosyalar oluşuyor.
#sistem uefi değilse **/sys/firmware/efi** konumunda dosyalar olmayacaktır.
$ if [[ -d /sys/firmware/efi ]] ; then
    mount --bind /sys/firmware/efi/efivars /target/sys/firmware/efi/efivars
fi

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

Şimdi de uefi kullandığımız için efivar bağlayalım. .. code-block:: shell

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

Sistem Kurulumu

Grub Kuralım

```
# 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 yapılandırması

1. /boot bölümünde initrd.img-<çekirdek-sürümü> dosyamızın olduğundan emin olalım.
2. /boot bölümünde vmlinuz-<çekirdek-sürümü> 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="4096" TYPE="ext4" PARTUUID="..."
```

Şimdi aşağıdaki gibi bir yapılandırma dosyası yazalım ve /boot/grub/grub.cfg dosyasına kaydedelim. Burada uuid değerini ve çekirdek sürümünü düzenleyin.

```
linux /vmlinuz-<çekirdek-sürümü>          root=UUID=<uuid-değeri> rw quiet  
initrd /initrd.img-<çekirdek-sürümü>  
boot
```

Ayrıca otomatik yapılandırma da oluşturabiliriz.

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

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	

Not: Disk bölümü konumu yerine **UUID=<uuid-değeri>** şeklinde yazmanızı öneririm. Bölüm adları değişebilirken uuid değerleri değişmez.

Uefi Sistem Kurulumu

Bu bölümde **Ext4** dosya sistemine grub kullanarak kurulum anlatılacaktır. Anlatım boyunca **/dev/sda** diski üzerinden örneklemeye yapılmıştır. Siz kendi diskinize göre düzenleyebilirsiniz. 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 modülleri **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.

Sistem Kurulumu

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

Şimdi de disk bölümümüzü bağlayalım.

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

Ardından dosyaları kopyalayalım.

```
# -p dosya izinlerini korur
# -r alt dizinlerle beraber kopyalar
# -f soru sormayı kapatır
# -v detaylı çıktıları gösterir
$ cp -prfv /source/* /target
```

Daha sonra diski senkronize edelim.

```
$ sync
```

Bootloader kurulumu

grub kurulumu yapmak için grub paketini kurulu olduğundan emin olun.

```
$ mkdir -p /target/dev
$ mkdir -p /target/sys
$ mkdir -p /target/proc
$ mkdir -p /target/run
$ mkdir -p /target/tmp
$ mount --bind /dev /target/dev
$ mount --bind /sys /target/sys
$ mount --bind /proc /target/proc
$ mount --bind /run /target/run
$ mount --bind /tmp /target/tmp
#efi alan bağlanıyor. Eğer uefi aktif edilmişse kernel */sys/firmware/efi** tarafından budizinler ve dosyalar oluşuyor.
#sistem uefi değilse */sys/firmware/efi** konumunda dosyalar olmayacaktır.
$ if [[ -d /sys/firmware/efi ]] ; then
    mount --bind /sys/firmware/efi/efivars /target/sys/firmware/efi/efivars
fi

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

Şimdi de uefi kullandığımız için efivar bağlayalım. .. code-block:: shell

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

Sistem Kurulumu

Grub Kuralım

```
# 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 yapılandırması

1. /boot bölümünde initrd.img-<çekirdek-sürümü> dosyamızın olduğundan emin olalım.
2. /boot bölümünde vmlinuz-<çekirdek-sürümü> 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="4096" TYPE="ext4" PARTUUID="..."
```

Şimdi aşağıdaki gibi bir yapılandırma dosyası yazalım ve /boot/grub/grub.cfg dosyasına kaydedelim. Burada uuid değerini ve çekirdek sürümünü düzenleyin.

```
linux /vmlinuz-<çekirdek-sürümü>          root=UUID=<uuid-değeri> rw quiet  
initrd /initrd.img-<çekirdek-sürümü>  
boot
```

Ayrıca otomatik yapılandırma da oluşturabiliriz.

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

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	

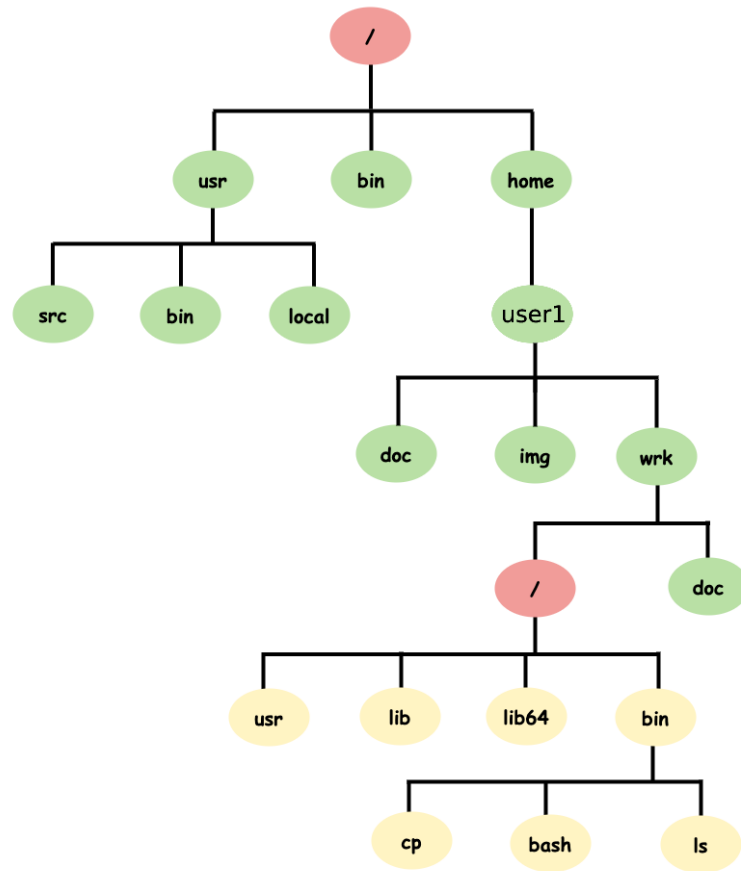
Not: Disk bölümü konumu yerine **UUID=<uuid-değeri>** şeklinde yazmanızı öneririm. Bölüm adları değişebilirken uuid değerleri değişmez.

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
- Sistem üzerinde yeni dağıtımlara müdahale etme ve sorun çözme
- Kullanıcı kendine özel geliştirme ortamı oluşturabilir.
- Yazılım bağımlılıkları 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/user1/wrk dizinindeki sistem üzerinde sisteme erişmek için;

```
sudo chroot /home/user1/wrk #sisteme erişim yapıldı.
```

/home/user1/wrk dizinindeki sistem üzerinde sistemi silmek için;

```
sudo rm -rf /home/user1/wrk #sistem silindi
```


Yardımcı Konular

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.

```
etapadmin@etahta: ~  
Dosya Düzenle Görünüm Ara Uçbirim Yardım  
etapadmin@etahta:~$ ldd /bin/ls  
linux-vdso.so.1 (0x00007ffc68471000)  
libgtk3-nocsd.so.0 => /usr/lib/x86_64-linux-gnu/libgtk3-nocsd.so.0 (0x00007ff3fa9db000)  
libselinux.so.1 => /lib/x86_64-linux-gnu/libselinux.so.1 (0x00007ff3fa9ad000)  
libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007ff3fa7cc000)  
libdl.so.2 => /lib/x86_64-linux-gnu/libdl.so.2 (0x00007ff3fa7c7000)  
libpthread.so.0 => /lib/x86_64-linux-gnu/libpthread.so.0 (0x00007ff3fa7c2000)  
libpcre2-8.so.0 => /usr/lib/x86_64-linux-gnu/libpcre2-8.so.0 (0x00007ff3fa726000)  
/lib64/ld-linux-x86-64.so.2 (0x00007ff3faa2a000)  
etapadmin@etahta:~$
```

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

Iddscript.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.

Yardımcı Konular

Sistem Dizinin Oluşturulması

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

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

ls Komutu

```
bash lddscripts.sh /bin/ls /home/etapadmin/test/ #komutu ile ls komutunu ve bağımlılığını kopyalandı.
```

```
etapadmin@etahta: ~  
Dosya Düzenle Görünüm Ara Uçbirim Yardım  
etapadmin@etahta:~$ bash lddscripts.sh /bin/ls /home/etapadmin/test/  
--> copy binary itself  
/bin -> /home/etapadmin/test/bin  
'/bin/ls' -> '/home/etapadmin/test/bin/ls'  
--> copy libraries  
/usr -> /home/etapadmin/test/usr  
/usr/lib -> /home/etapadmin/test/usr/lib  
/usr/lib/x86_64-linux-gnu -> /home/etapadmin/test/usr/lib/x86_64-linux-gnu  
'/usr/lib/x86_64-linux-gnu/libgtk3-nocsd.so.0' -> '/home/etapadmin/test/usr/lib/x86_64-linux-gnu/libgtk3-nocsd.so.0'  
/lib -> /home/etapadmin/test/lib  
/lib/x86_64-linux-gnu -> /home/etapadmin/test/lib/x86_64-linux-gnu  
'/lib/x86_64-linux-gnu/libselinux.so.1' -> '/home/etapadmin/test/lib/x86_64-linux-gnu/libselinux.so.1'  
'/lib/x86_64-linux-gnu/libc.so.6' -> '/home/etapadmin/test/lib/x86_64-linux-gnu/libc.so.6'  
'/lib/x86_64-linux-gnu/libdl.so.2' -> '/home/etapadmin/test/lib/x86_64-linux-gnu/libdl.so.2'  
'/lib/x86_64-linux-gnu/libpthread.so.0' -> '/home/etapadmin/test/lib/x86_64-linux-gnu/libpthread.so.0'  
'/usr/lib/x86_64-linux-gnu/libpcre2-8.so.0' -> '/home/etapadmin/test/usr/lib/x86_64-linux-gnu/libpcre2-8.so.0'  
/lib64 -> /home/etapadmin/test/lib64  
'/lib64/ld-linux-x86-64.so.2' -> '/home/etapadmin/test/lib64/ld-linux-x86-64.so.2'  
etapadmin@etahta:~$
```

Bu işlemi diğer komutlar içinde sırasıyla yapmamız gerekmektedir.

rmdir Komutu

```
bash lddscripts.sh /bin/rmdir /home/etapadmin/test/ #komutu ile rmdir komutunu ve bağımlılığını kopyalandı.
```

```
etapadmin@etahta: ~  
Dosya Düzenle Görünüm Ara Uçbirim Yardım  
etapadmin@etahta:~$ bash lddscripts.sh /bin/rmdir /home/etapadmin/test/  
--> copy binary itself  
'/bin/rmdir' -> '/home/etapadmin/test/bin/rmdir'  
--> copy libraries  
'/usr/lib/x86_64-linux-gnu/libgtk3-nocsd.so.0' -> '/home/etapadmin/test/usr/lib/x86_64-linux-gnu/libgtk3-nocsd.so.0'  
'/lib/x86_64-linux-gnu/libc.so.6' -> '/home/etapadmin/test/lib/x86_64-linux-gnu/libc.so.6'  
'/lib/x86_64-linux-gnu/libdl.so.2' -> '/home/etapadmin/test/lib/x86_64-linux-gnu/libdl.so.2'  
'/lib/x86_64-linux-gnu/libpthread.so.0' -> '/home/etapadmin/test/lib/x86_64-linux-gnu/libpthread.so.0'  
'/lib64/ld-linux-x86-64.so.2' -> '/home/etapadmin/test/lib64/ld-linux-x86-64.so.2'  
etapadmin@etahta:~$
```

Yardımcı Konular

mkdir Komutu

`bash lddscripts.sh /bin/mkdir /home/etapadmin/test/` #komutu ile mkdir komutunu ve bağımlılığını kopyalandı.

```
etapadmin@etahta: ~  
Dosya Düzenle Görünüm Ara Uçbirim Yardım  
etapadmin@etahta:~$ bash lddscripts.sh /bin/mkdir /home/etapadmin/test/  
---> copy binary itself  
'/bin/mkdir' -> '/home/etapadmin/test/bin/mkdir'  
---> copy libraries  
'/usr/lib/x86_64-linux-gnu/libgtk3-nocsd.so.0' -> '/home/etapadmin/test/usr/lib/x86_64-linux-gnu/libgtk3-nocsd.so.0'  
'/lib/x86_64-linux-gnu/libselinux.so.1' -> '/home/etapadmin/test/lib/x86_64-linux-gnu/libselinux.so.1'  
'/lib/x86_64-linux-gnu/libc.so.6' -> '/home/etapadmin/test/lib/x86_64-linux-gnu/libc.so.6'  
'/lib/x86_64-linux-gnu/libdl.so.2' -> '/home/etapadmin/test/lib/x86_64-linux-gnu/libdl.so.2'  
'/lib/x86_64-linux-gnu/libpthread.so.0' -> '/home/etapadmin/test/lib/x86_64-linux-gnu/libpthread.so.0'  
'/usr/lib/x86_64-linux-gnu/libpcre2-8.so.0' -> '/home/etapadmin/test/usr/lib/x86_64-linux-gnu/libpcre2-8.so.0'  
'/lib64/ld-linux-x86-64.so.2' -> '/home/etapadmin/test/lib64/ld-linux-x86-64.so.2'  
etapadmin@etahta:~$
```

bash Komutu

`bash lddscripts.sh /bin/bash /home/etapadmin/test/` #komutu ile bash komutunu ve bağımlılığını kopyalandı.

```
etapadmin@etahta: ~  
Dosya Düzenle Görünüm Ara Uçbirim Yardım  
etapadmin@etahta:~$ bash lddscripts.sh /bin/bash /home/etapadmin/test/  
---> copy binary itself  
'/bin/bash' -> '/home/etapadmin/test/bin/bash'  
---> copy libraries  
'/usr/lib/x86_64-linux-gnu/libgtk3-nocsd.so.0' -> '/home/etapadmin/test/usr/lib/x86_64-linux-gnu/libgtk3-nocsd.so.0'  
'/lib/x86_64-linux-gnu/libtinfo.so.6' -> '/home/etapadmin/test/lib/x86_64-linux-gnu/libtinfo.so.6'  
'/lib/x86_64-linux-gnu/libc.so.6' -> '/home/etapadmin/test/lib/x86_64-linux-gnu/libc.so.6'  
'/lib/x86_64-linux-gnu/libdl.so.2' -> '/home/etapadmin/test/lib/x86_64-linux-gnu/libdl.so.2'  
'/lib/x86_64-linux-gnu/libpthread.so.0' -> '/home/etapadmin/test/lib/x86_64-linux-gnu/libpthread.so.0'  
'/lib64/ld-linux-x86-64.so.2' -> '/home/etapadmin/test/lib64/ld-linux-x86-64.so.2'  
etapadmin@etahta:~$
```

chroot Sistemde Çalışma

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

```
etapadmin@etahta: ~  
Dosya Düzenle Görünüm Ara Uçbirim Yardım  
etapadmin@etahta:~$ sudo chroot /home/etapadmin/test  
[sudo] password for etapadmin:  
bash-5.2# ls  
bin lib lib64 usr  
bash-5.2# mkdir abc  
bash-5.2# ls  
abc bin lib lib64 usr  
bash-5.2# rmdir abc  
bash-5.2# ls  
bin lib lib64 usr  
bash-5.2# pwd  
/  
bash-5.2# ldd  
bash: ldd: command not found  
bash-5.2# exit  
exit  
etapadmin@etahta:~$
```

- **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>

Qemu Kullanımı



Qemu Nedir?

Açık kaynaklı sanallaştırma aracıdır.

Kaynak dosyalarından kurulum için;

```
git clone https://gitlab.com/qemu-project/qemu.git
cd qemu
./configure
make
sudo make install
```

Sisteme Kurulum

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

- 30GB bir disk oluşturup etahta.iso dosyamızı 2GB ramdan oluşan bir makina çalıştıralım.

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

- Oluşturulan sanal disk ve 2GB ram ile açma.

```
qemu-system-x86_64 --enable-kvm -hda disk.img -m 2G #sadece disk ile çalıştırılıyor
```

- Sistemi etahta.iso dosyamızı 2GB ramdan oluşan bir makina olarak çalıştıralım.

```
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 belirlemek için boot menu gelmesini istersek aşağıdaki gibi komut satırına seçenek eklemeliyiz.

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

Uefi kurulum için:

```
sudo apt-get install ovmf
```

Yardımcı Konular

```
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:

kendi ipsi:10.0.2.15

ana bilgisayar 10.0.0.2 olarak ayarlıyor.

vmlinuz ve initrd

Daha sonra da qemu kullanarak test edelim.

```
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. Bu bağlamda SquashFS dosya sistemi ve sıkıştırma nasıl yapılır bu dokümanda anlatılmaktadır.

SquashFS Nedir?

SquashFS, Linux işletim sistemlerinde sıkıştırılmış bir dosya sistemidir. Bu dosya sistemi, sıkıştırma algoritması kullanarak dosyaları sıkıştırır ve ardından salt okunur bir dosya sistemine dönüştürür. SquashFS, özellikle gömülü sistemlerde ve Linux dağıtımlarında kullanılan bir dosya sistemidir.

SquashFS Oluşturma

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

Cdrom Erişimi

/dev/sr0, Linux işletim sistemlerinde bir CD veya DVD sürücüsünü temsil eden bir aygıt dosyasıdır. Bu dosya, CD veya DVD sürücüsünün fiziksel cihazını temsil eder ve kullanıcıların bu sürücüye erişmesini sağlar.

/dev/sr0 dosyası, Linux'un aygıt dosyası sistemi olan /dev dizininde bulunur. Bu dosya, Linux çekirdeği tarafından otomatik olarak oluşturulur ve sürücüye bağlı olarak farklı bir isim alabilir. Örneğin, ikinci bir CD veya DVD sürücüsü /dev/sr1 olarak adlandırılabilir.

Bu aygıt dosyası, kullanıcıların CD veya DVD'leri okumasına veya yazmasına olanak tanır. Örneğin, bir CD'yi okumak için aşağıdaki gibi bir komut kullanabilirsiniz:

```
$ cat /dev/sr0
```

Cdrom Bağlama

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

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

squashfs Dosyasını Bulma

Genellikle isoların içine squashfs dosyası oluşturulur. Bu sayede live yükleme yapılabilir. Örneğin /live/filesystem.squashfs benim imajlarımda böyle konumlandırıyorum.

squashfs Bağlama

squashfs dosyasını bağlamadan önce loop modülünün yüklü olması gerekmektedir. eğer yüklemeyerseniz

```
modprobe loop #loop modülü yüklenir.
```

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

squashfs Sistemine Geçiş

Yukarıdaki adımlarda squashfs dosyamızı /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
```

Yardımcı Konular

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 çıkarttığımızda kernel derlemek ciddi bir iş yükü ve karmaşa yaratacaktır.

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

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

```
ln -s kmod /bin/depmod
ln -s kmod /bin/insmod
ln -s kmod /initrd/bin/lsmmod
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.

Modul Yazma

hello.c dosyamız

```
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/init.h>
MODULE_DESCRIPTION("Hello World examples");
MODULE_LICENSE("GPL");
MODULE_AUTHOR("Bayram");
static int __init hello_init(void)
{
    printk(KERN_INFO "Hello world!\n");
    return 0;
}
static void __exit hello_cleanup(void)
{
    printk(KERN_INFO "remove module.\n");
}
module_init(hello_init);
module_exit(hello_cleanup);
```


Yardımcı Konular

Makefile dosyamız

```
obj-m+=my_module.o
all:
    make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules
clean:
    make -C /lib/modules/$(shell uname -r)/build M=$(PWD) clean
```

modülün derlenmesi ve eklenip kaldırılması

```
make

insmod my_modul.ko // modül kernele eklendi.

lsmod | grep my_modul //modül yüklendi mi kontrol ediliyor.

rmmod my_modul // modül kernelden çıkartılıyor.
```

Not:

dmesg ile log kısmında eklendiğinde Hello Word yazısını ve kaldırıldığında modul ismini görebiliriz.

sfdisk Nedir

sfdisk, Linux işletim sistemlerinde disk bölümlerini yönetmek için kullanılan bir komuttur. Disk bölümlerini oluşturmak, düzenlemek, silmek veya görüntülemek için sfdisk'i kullanabilirsiniz.

Disk Bölümlerini Görüntüleme:

Diskinizdeki mevcut bölümleri görüntülemek için sfdisk komutunu kullanabilirsiniz. Aşağıdaki komutu kullanarak mevcut bölümleri listeleyebilirsiniz:

```
sfdisk -l /dev/sda
```

Disk Bölümleri Oluşturma:

Yeni bir disk bölümü oluşturmak için sfdisk komutunu kullanabilirsiniz. Örneğin, /dev/sda üzerinde yeni bir bölüm oluşturmak için aşağıdaki komutu kullanabilirsiniz:

```
echo ",,L" | sfdisk /dev/sda
```

Bu komut, kullanılabilir tüm alanı kullanarak bir bölüm oluşturacaktır.

Disk Bölümlerini Silme:

Bir disk bölümünü silmek için sfdisk komutunu kullanabilirsiniz. Örneğin, /dev/sda üzerindeki bir bölümü silmek için aşağıdaki komutu kullanabilirsiniz:

```
echo ",,L" | sfdisk --delete /dev/sda
```

Bu komut, belirtilen bölümü silecektir.

sfdisk komutunun daha fazla seçeneği ve kullanımı vardır. Daha fazla bilgi için sfdisk komutunun man sayfasını inceleyebilirsiniz:

Disk Etiketini Belirleme

```
echo 'label: dos' | sfdisk /dev/vda #dos label  
echo 'label: gpt' | sfdisk /dev/vda #gpt label
```

Bazı Örnekler

Örnek1:

```
sfdisk /dev/sdf <<EOF  
0,512  
,512  
;  
EOF  
  
/dev/sdf1 0+ 511 512- 4112639+ 83 Linux  
/dev/sdf2 512 1023 512 4112640 83 Linux  
/dev/sdf3 1024 1043 20 160650 83 Linux
```

Örnek2:

```
sfdisk /dev/vda << EOF
label: dos
label-id: 0xaaaaaaaa
# comment:
start=, size= 50M, type= 7 , bootable
start=, size= 650M, type= 27
start=, size= 45G , type= 7
EOF
```

Örnek3:

Bu örnekte ilk bölüm 1GB ve ikinci bölüm ise diskin geri kalan kısmıdır.

```
echo -e "label: gpt\n,1GiB\n," | sudo sfdisk /dev/vda
veya
sfdisk /dev/vda << EOF
label: gpt
,1GiB
,
EOF
```

Örnek4:

```
my.layout
# partition table of /dev/sda
unit: sectors

/dev/sda1 : start=    2048, size=   497664, Id=83, bootable
/dev/sda2 : start=   501758, size=1953021954, Id= 5
/dev/sda3 : start=      0, size=      0, Id= 0
/dev/sda4 : start=      0, size=      0, Id= 0
/dev/sda5 : start=   501760, size=1953021952, Id=8e
```

Aynı disk bölümlemesini ve düzenini başka bir aygıtı uygulamak için:

```
sfdisk /dev/sdb < my.layout
```

İmza Doğrulama

GPG (GNU Privacy Guard), dosyaların ve iletişimin güvenliğini sağlamak için kullanılan bir şifreleme aracıdır. Bu araçla, dosyaları şifreleyebilir, imzalayabilir ve imzaları doğrulayabiliriz.

İmza Oluşturma

GPG ile anahtar oluşturmak oldukça basittir. İşte adım adım nasıl yapılacağı: İlk olarak, GPG yazılımını sisteminize yüklemeniz gerekmektedir. Linux tabanlı bir işletim sistemi kullanıyorsanız, terminali açın ve aşağıdaki komutu çalıştırın ve GPG kurulumunu yapınız.

language-bash

```
sudo apt-get install gnupg
```

GPG anahtar çiftini oluşturmak için aşağıdaki komutu kullanın:

language-bash

```
gpg --full-generate-key
```

Belge İmzalama

Anahtar çifti oluşturulduktan sonra, imzalamak istediğiniz belgeyi seçin ve aşağıdaki komutu kullanarak belgeyi imzalayın:

language-bash

```
gpg --sign belge.txt
```

1. İmzalanan belge, aynı dizinde "belge.txt.asc" uzantısıyla kaydedilecektir. Bu imzalı belgeyi başkalarıyla paylaşabilirsiniz.

İmzalı Belge Doğrulama

- İmzalı belgeyi doğrulamak istediğinizde, aşağıdaki komutu kullanarak GPG'yi kullanabilirsiniz:

language-bash

```
gpg --verify belge.txt.asc
```

Bu komut, belgenin doğruluğunu kontrol edecek ve imzanın geçerli olup olmadığını size bildirecektir. İmza doğrulama işlemleri daha detaylı bir şekilde aşağıda anlatılmıştır.

bash ile Doğrulama

bash script ile imza doğrulaması aşağıdaki kodlarla yapılabilir.

```
#!/bin/bash
file=$1
if [ $file == "" ]
then
echo "Dosya belirtiniz."
else
    gpg --verify "$file"
    status=$?
    if [ "$status" == "0" ]
    then
        echo "İmza İyi"
    else
        echo "İmza Kötü"
```

fi

```

belge.txt.asc (~/.Masaüstü/gpg) - gedit
1 ----BEGIN PGP SIGNED MESSAGE-----
2 Hash: SHA512
3
4 hava güneşli
5 ----BEGIN PGP SIGNATURE-----
6
7 iLMEAEKABOWIOU+W8kMxqjL6zWqt8c20LzB0WrwUCZwt4YQAKRB8c
8 r2amBACBumERXzB1B19EgZMJR4+Mbevk97qk3FRKVbp8T/
9 zml187IxuGdNENdVUK
9 snJftSRVzgsC/OQ2tnhGM3be2gGqM4aFI4J0m41G3JV+L+An/
DmTwk6q8t8e8/L
10 oGgiED9KCJ/Q5/DqE2OM0ePfwCJpet5SY0KUL5PwYnZBRPnABA==
11 =T817
12 -----END PGP SIGNATURE-----

imzadogrula.sh (~/.Masaüstü/gpg) - gedit
1 #!/bin/bash
2 file=$1
3 if [ $file == "" ]
4 then
5 echo "Dosya belirtiniz."
6 else
7 gpg --verify "$file"
8 status=$?
9 if [ "$status" == "0" ]
10 then
11 echo "İmza İyi"
12 else
13 echo "İmza Kötü"
14 fi
15 #!

belge.txt.asc (~/.Masaüstü/gpg) - gedit
1 ----BEGIN PGP SIGNED MESSAGE-----
2 Hash: SHA512
3
4 hava güneşli
5 ----BEGIN PGP SIGNATURE-----
6
7 iLMEAEKABOWIOU+W8kMxqjL6zWqt8c20LzB0WrwUCZwt4YQAKRB8c
8 r2amBACBumERXzB1B19EgZMJR4+Mbevk97qk3FRKVbp8T/
9 zml187IxuGdNENdVUK
9 snJftSRVzgsC/OQ2tnhGM3be2gGqM4aFI4J0m41G3JV+L+An/
DmTwk6q8t8e8/L
10 oGgiED9KCJ/Q5/DqE2OM0ePfwCJpet5SY0KUL5PwYnZBRPnABA==
11 =T817
12 -----END PGP SIGNATURE-----

imzadogrula.sh (~/.Masaüstü/gpg) - gedit
1 #!/bin/bash
2 file=$1
3 if [ $file == "" ]
4 then
5 echo "Dosya belirtiniz."
6 else
7 gpg --verify "$file"
8 status=$?
9 if [ "$status" == "0" ]
10 then
11 echo "İmza İyi"
12 else
13 echo "İmza Kötü"
14 fi
15 #!

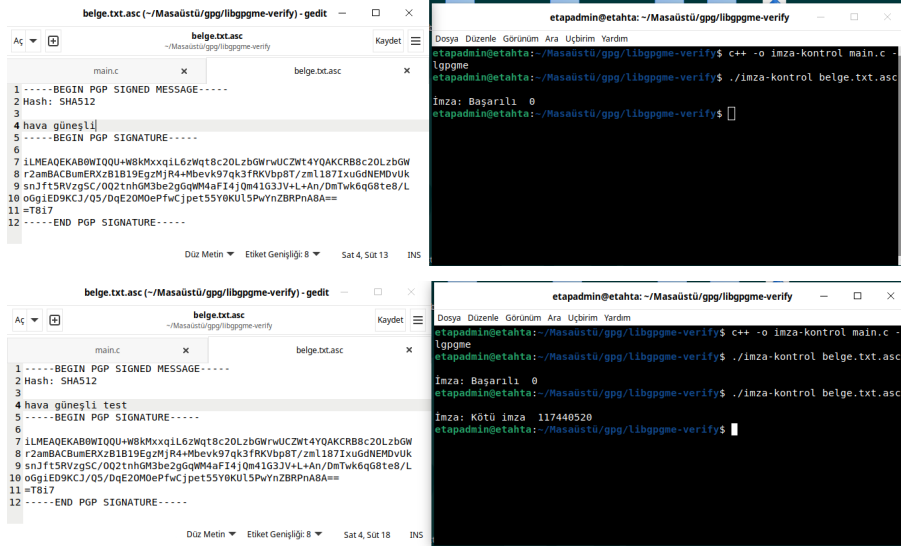
etapadmin@etahta: ~/.Masaüstü/gpg
Dosya Düzenle Görünüm Ara Uçbirim Yardım
etapadmin@etahta:~/.Masaüstü/gpg$ bash imzadogrula.sh belge.txt.asc
gpg: İmza Cts 02 Ara 2023 21:33:05 +03 de
gpg: RSA kullanılarak anahtar 14F96F24331C6A88BEB35AAB7C73638BCDB196A
F ile yapılmış
gpg: "karahan <bayram@gmail.com>" deki imza iyi [bilinmeyen]
gpg: UYARI: Bu anahtar güven dereceli bir imza ile sertifikalanmamış!
gpg: Bu imzanın sahibine ait olduğuna dair bir belirti yok.
Birincil anahtar parmak izi: 14F9 6F24 331C 6A88 BEB3 5AAB 7C73 638B CDB1 96AF
gpg: WARNING: not a detached signature; file 'belge.txt' was NOT verified!
İmza İyi
etapadmin@etahta:~/.Masaüstü/gpg$

```

Yardımcı Konular

c++ ile Doğrulama

c kullanarak özünde bash komutunu sonucunu kontrol eden imza doğrulaması aşağıdaki kodlarla yapılabilir.



```
belge.txt.asc (~/.Masaüstü/gpg/libpgpgme-verify) - gedit
Açık | Belge.txt.asc | Kaydet
main.c | belge.txt.asc
1 -----BEGIN PGP SIGNED MESSAGE-----
2 Hash: SHA512
3
4 hava güneşli
5 -----BEGIN PGP SIGNATURE-----
6
7 lLMEAEKABOWIQU+WBkMxxq1L6zWqt8c20LzbGwruJCZt4YQAKCRB8c20LzbGw
8 r2ambACumERXzB1B19EgZMjR4+Mbevk97qk3fRKVbp8T/zml187IuGdNEMDVUK
9 snJft5RVZg5C/0QZtnhG3be2GqW4aF14jQm41G3JV+L+An/DmTwk6qG8te8/L
10 oGg1ED9KCj/Q5/DqE2OM0ePfwCjpet55Y8KUL5PwYnZBRPhA8A==
11 =T817
12 -----END PGP SIGNATURE-----
Düz Metin | Etiket Genişliği: 8 | Sat 4, Süt 13 | INS

etapadmin@etahta: ~/.Masaüstü/gpg/libpgpgme-verify
Dosya Düzenle Görünüm Ara Uçbirim Yardım
etapadmin@etahta: ~/.Masaüstü/gpg/libpgpgme-verify$ c++ -o imza-kontrol main.c -lgpgme
etapadmin@etahta: ~/.Masaüstü/gpg/libpgpgme-verify$ ./imza-kontrol belge.txt.asc
Imza: Başarılı 0
etapadmin@etahta: ~/.Masaüstü/gpg/libpgpgme-verify$

etapadmin@etahta: ~/.Masaüstü/gpg/libpgpgme-verify
etapadmin@etahta: ~/.Masaüstü/gpg/libpgpgme-verify$ c++ -o imza-kontrol main.c -lgpgme
etapadmin@etahta: ~/.Masaüstü/gpg/libpgpgme-verify$ ./imza-kontrol belge.txt.asc
Imza: Başarılı 0
etapadmin@etahta: ~/.Masaüstü/gpg/libpgpgme-verify$ ./imza-kontrol belge.txt.asc
Imza: Kötü imza 117440520
etapadmin@etahta: ~/.Masaüstü/gpg/libpgpgme-verify$
```

```
#include <iostream> #include <cstdlib>
```

```
int main() {
```

```
    int result = system("gpg --verify belge.txt.asc"); if (result == 0) {
```

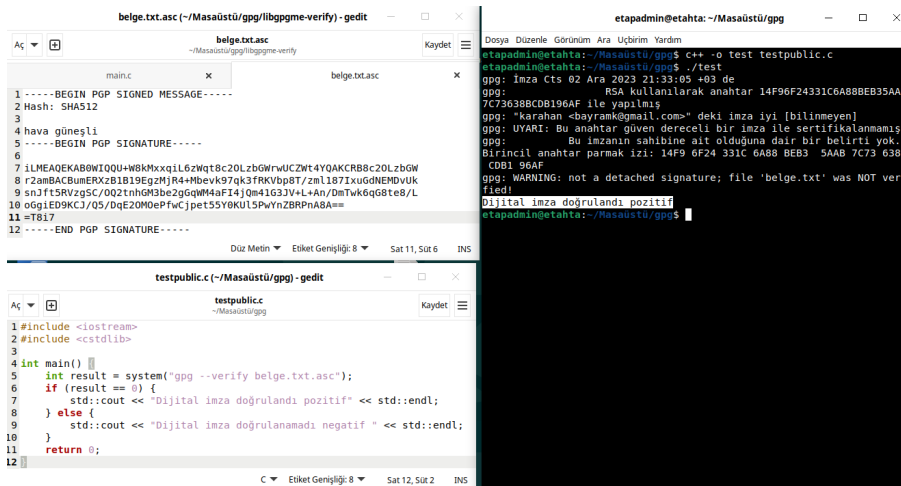
```
        std::cout << "Dijital imza doğrulandı pozitif" << std::endl;
```

```
    } else {
```

```
        std::cout << "Dijital imza doğrulanamadı negatif " << std::endl;
```

```
    } return 0;
```

```
}
```

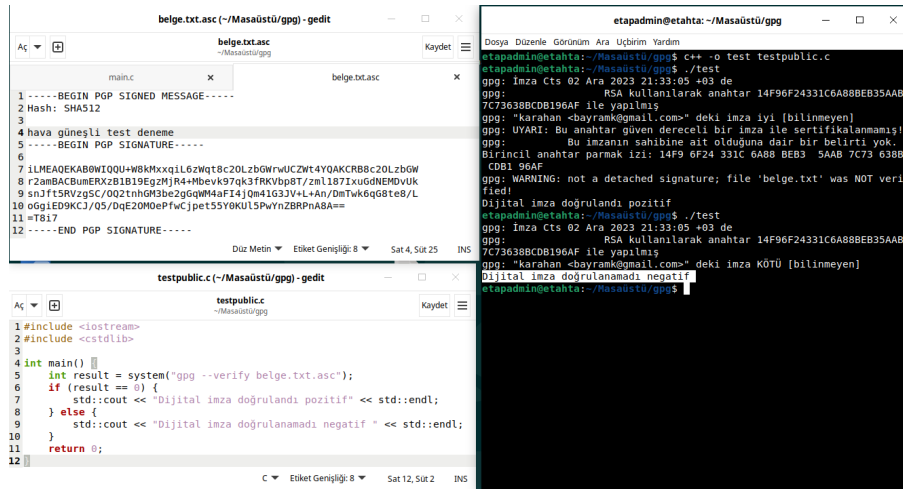


```
belge.txt.asc (~/.Masaüstü/gpg/libpgpgme-verify) - gedit
Açık | Belge.txt.asc | Kaydet
main.c | belge.txt.asc
1 -----BEGIN PGP SIGNED MESSAGE-----
2 Hash: SHA512
3
4 hava güneşli
5 -----BEGIN PGP SIGNATURE-----
6
7 lLMEAEKABOWIQU+WBkMxxq1L6zWqt8c20LzbGwruJCZt4YQAKCRB8c20LzbGw
8 r2ambACumERXzB1B19EgZMjR4+Mbevk97qk3fRKVbp8T/zml187IuGdNEMDVUK
9 snJft5RVZg5C/0QZtnhG3be2GqW4aF14jQm41G3JV+L+An/DmTwk6qG8te8/L
10 oGg1ED9KCj/Q5/DqE2OM0ePfwCjpet55Y8KUL5PwYnZBRPhA8A==
11 =T817
12 -----END PGP SIGNATURE-----
Düz Metin | Etiket Genişliği: 8 | Sat 11, Süt 6 | INS

testpublic.c (~/.Masaüstü/gpg) - gedit
Açık | testpublic.c | Kaydet
1 #include <iostream>
2 #include <cstdlib>
3
4 int main() {
5     int result = system("gpg --verify belge.txt.asc");
6     if (result == 0) {
7         std::cout << "Dijital imza doğrulandı pozitif" << std::endl;
8     } else {
9         std::cout << "Dijital imza doğrulanamadı negatif " << std::endl;
10    }
11    return 0;
12 }
C | Etiket Genişliği: 8 | Sat 12, Süt 2 | INS

etapadmin@etahta: ~/.Masaüstü/gpg
Dosya Düzenle Görünüm Ara Uçbirim Yardım
etapadmin@etahta: ~/.Masaüstü/gpg$ c++ -o test testpublic.c
etapadmin@etahta: ~/.Masaüstü/gpg$ ./test
gpg: Imza Cts 02 Ara 2023 21:33:05 +03 de
gpg: 7C72638BCDB196AF ile yapılmış
gpg: "karanan <bayram@gmail.com>" deki imza iyi [bilinmeyen]
gpg: UYARI: Bu anahtar güven dereceli bir imza ile sertifikalanmamış!
gpg: Bu imzanın sahibine ait olduğuna dair bir belirti yok.
gpg: Birincil anahtar parmak izi: 14F9 6F24 331C 6A88 BEB3 5AAB 7C73 638B CDB1 96AF
gpg: WARNING: not a detached signature; file 'belge.txt' was NOT veri
fied!
Dijital imza doğrulandı pozitif
etapadmin@etahta: ~/.Masaüstü/gpg$
```

Yardımcı Konular



```
belge.txt.asc (~/.Masaüstü/gpg) - gedit
main.c x belge.txt.asc x
1 -----BEGIN PGP SIGNED MESSAGE-----
2 Hash: SHA512
3
4 hava güneşli test deneme
5 -----BEGIN PGP SIGNATURE-----
6
7 iLMEAOEKAB0W1QOU+H8kMxqiL6zwt8c20LzbGWwUCZwt4Y0AKCRB8c20LzbGW
8 r2anBACBumERXz81B19EgZMjR4+HbeV97q3fRKVbp8T/zm1187IXuGdNEMDVUk
9 snJftSRVz9Sc/QQ2tnhGm3be2G6qM4aFI4j0m41G3JV+L+An/DmTwk6qG8te8/L
10 oGgiED9KCJ/05/DqE20M0ePfwCjpet5SY0KUL5PwYnZBRPnA8A==
11 =T817
12 -----END PGP SIGNATURE-----
Düz Metin Etiket Geniği: 8 Sat 4, Süt 25 INS

testpublic.c (~/.Masaüstü/gpg) - gedit
1 #include <iostream>
2 #include <cstdlib>
3
4 int main() {
5     int result = system("gpg --verify belge.txt.asc");
6     if (result == 0) {
7         std::cout << "Dijital imza doğrulandı pozitif" << std::endl;
8     } else {
9         std::cout << "Dijital imza doğrulanamadı negatif" << std::endl;
10    }
11    return 0;
12 }
C Etiket Geniği: 8 Sat 12, Süt 2 INS

etapadmin@etahta: ~/.Masaüstü/gpg
Dosya Düzenle Görünüm Ara Uçbirim Yardım
etapadmin@etahta: ~/.Masaüstü/gpg$ c++ -o test testpublic.c
etapadmin@etahta: ~/.Masaüstü/gpg$ ./test
gpg: Imza Cts 02 Ara 2023 21:33:05 +03 de
gpg:
gpg: RSA kullanılarak anahtar 14F96F24331C6A88BE835AAB
7C73638BCDB196AF ile yapılmış
gpg: "karahan <bayramk@gmail.com>" deki imza iyi [bilinmeyen]
gpg: UYARI: Bu anahtar güven dereceli bir imza ile sertifikalanmamış!
gpg: Bu imzanın sahibine ait olduğuna dair bir belirti yok.
Birincil anahtar parmak izi: 14F9 6F24 331C 6A88 BEB3 5AAB 7C73 638B
CDB1 96AF
gpg: WARNING: not a detached signature; file 'belge.txt' was NOT veri
fied!
Dijital imza doğrulandı pozitif
etapadmin@etahta: ~/.Masaüstü/gpg$ ./test
gpg: Imza Cts 02 Ara 2023 21:33:05 +03 de
gpg:
gpg: RSA kullanılarak anahtar 14F96F24331C6A88BE835AAB
7C73638BCDB196AF ile yapılmış
gpg: "karahan <bayramk@gmail.com>" deki imza KÖTÜ [bilinmeyen]
Dijital imza doğrulanamadı negatif
etapadmin@etahta: ~/.Masaüstü/gpg$
```

Yardımcı Konular

c++ libpgpme ile Doğrulama

libpgpme kütüphanelerini kullanarak bir belge doğrulama yapabiliriz.

```
#include <stdio.h>
#include <gpgme.h>
#include <locale.h>
#include <stdlib.h>
#include <string.h>

int print_engine_info() {
    gpgme_engine_info_t info;
    gpgme_error_t err;

    err = gpgme_get_engine_info(&info);
    if (err != GPG_ERR_NO_ERROR) {
        fprintf(stderr, "ERROR: Filed to get engine info!\n");
        return -1;
    }
    printf( "Installed engines: {\n" );
    while(info != NULL) {
        printf( "\t* %s Protocol=%s Version=%s Required-Version=%s Home=%s\n",
            info->file_name, gpgme_get_protocol_name(info->protocol),
            info->version, info->req_version, info->home_dir );
        info = info->next;
    }
    printf("}\n");
    return 0;
}

int main(int argc, const char* argv[]) {
    const char *gpgme_version, *gpgme_prot;
    gpgme_error_t err;
    gpgme_ctx_t ctx;
    FILE *fp_sig=NULL, *fp_msg=NULL;
    gpgme_data_t sig=NULL, msg=NULL, plain=NULL, text=NULL;
    gpgme_verify_result_t result;

    gpgme_protocol_t protocol = GPGME_PROTOCOL_OpenPGP;

    /* GPGME version check and initialization */
    setlocale(LC_ALL, "");

    gpgme_version = gpgme_check_version(GPGME_VERSION);    // developed for 1.5.1
    if (!gpgme_version) {
        fprintf(stderr, "ERROR: Wrong library on target! Please "
            "install at least version %s!\n", GPGME_VERSION);
        exit(1);
    }
    gpgme_set_locale(NULL, LC_CTYPE, setlocale(LC_CTYPE, NULL));
#ifdef LC_MESSAGES
    gpgme_set_locale(NULL, LC_MESSAGES, setlocale(LC_MESSAGES, NULL));
#endif
}
```

```
/* Protocol check */
gpgme_prot = gpgme_get_protocol_name(protocol);
err = gpgme_engine_check_version(protocol);
if (!gpgme_prot || err != GPG_ERR_NO_ERROR) {
    fprintf(stderr, "ERROR: libpgpme lacks of OpenPGP protocol!\n");
    print_engine_info();
    exit(1);
}

fp_sig = fopen(argv[1], "rb");
if (!fp_sig) {
```

```
    fprintf(stderr, "ERROR: Failed to open '%s'!\n", argv[0]);
    exit(1);
}
```



```

    }
    if (argc > 2)
    {
        fp_msg = fopen(argv[2], "rb");
        if (!fp_msg)
        {
            fprintf(stderr, "ERROR: Failed to open '%s'!\n", argv[1]);
            exit(1);
        }
    }

    err = gpgme_new(&ctx);
    if (err != GPG_ERR_NO_ERROR) {
        char buf[4096];
        gpgme_strerror_r(err, buf, 4096);
        fprintf(stderr, "ERROR: %s\n", buf);
        exit(1);
    }

    gpgme_set_protocol(ctx, protocol);

    err = gpgme_data_new_from_stream(&sig, fp_sig);
    if (err) {
        fprintf(stderr, "ERROR allocating data object: %s\n", gpgme_strerror(err));
        exit(1);
    }

    if (fp_msg)
    {
        err = gpgme_data_new_from_stream(&msg, fp_msg);
        if (err) {
            fprintf(stderr, "ERROR allocating data object: %s\n", gpgme_strerror(err));
            exit(1);
        }
        printf("Loaded message from '%s'\n", argv[2]);
    }
    else
    {
        err = gpgme_data_new(&plain);
        if (err) {
            fprintf(stderr, "ERROR allocating data object: %s\n", gpgme_strerror(err));
            exit(1);
        }
        ///printf("Allocated 'plain' data\n");
    }

    err = gpgme_op_verify(ctx, sig, msg, plain);
    if (err)
    {
        fprintf(stderr, "ERROR: signing failed: %s\n", gpgme_strerror(err));
        exit(1);
    }

    result = gpgme_op_verify_result(ctx);
    int count = 0;

```

```

    if (result) {
        gpgme_signature_t sig;

        for(sig = result->signatures; sig; sig = sig->next)
        {
            count += 1;
            if ( !(sig->summary & GPGME_SIGSUM_VALID) ) {
                printf("İmza: %s %d\n", gpgme_strerror(sig->status), sig->status);

                exit(1);
            }
        }
    }

```

```

    }

```

```
if (count < 1) {
    printf( "Error: Cannot find matching signature!\n" );
    return 1;
}

printf( "\nSignature verification successful. Plaintext:\n" );

text = plain ? plain : msg;
gpgme_data_seek(text, 0, SEEK_SET);
size_t bytes;
do {
    char buffer[256];
    bytes = gpgme_data_read(text, buffer, 256-1);
    buffer[bytes] = '\0';

    printf( "%s", buffer );
} while( bytes > 0 );

gpgme_data_release(plain);
gpgme_data_release(msg);
gpgme_data_release(sig);

gpgme_release(ctx);

return 0;
}
```

Kernel Modul Derleme

Kernel linux sistemlerinin temel dosyasıdır.

Kaynak Dosya İndirme

Linux çekirdeğinin kaynak kodunu <https://kernel.org> üzerinden indirin.

```
wget https://cdn.kernel.org/pub/linux/kernel/v6.x/linux-6.6.10.tar.xz
tar -xvf linux-6.6.10.tar.xz
cd linux-6.6.10
```

Kernel Ayarları

Arşiv açıldıktan sonra arşivin açıldığı dizinde **.config** dosyası oluşturmamızdır. Bu dosyada hangi ayarlara göre derleme yapılacağını belirten parametreler var. Eğer temel(varsayılan) ayarlarda derlenmesini istersek **make defconfig** komutuyla **.config** adında bir dosya oluşturacaktır.

Ben kendim belirleyeceğim bu ayarları diyorsak **make menuconfig** komutuyla açılan ekrandan ayarlamaları yapıp ayarları kaydetmeliyiz. Ayarlar kaydedilince **.config** dosyası oluşacaktır.

Bunların dışında ben Arch, Debian vb. dağıtımların **.config** dosyasını kullanacağım da diyebilirsiniz. Burada Arch Linux **.config** dosyasını kullanacağız.

https://gitlab.archlinux.org/archlinux/packaging/packages/linux/-/blob/main/config?ref_type=heads

Bu adresten indirilen **config** dosyasını **.config** olarak tarball(tar uzantılı sıkıştırılmış) dosyasının açıldığı dizine kopyalayalım.

Kernel Derleme

.config ayarlamaları yapıldıktan sonra derleme yapılacak. Derleme aşağıdaki komutla yapılır.

```
make bzImage # tek çekirdekle derleme yapacak yavaş olur
#make bzImage -j$(proc) #işlemci çekirdek sayısını sistemden alarak yapıyor, hızlı olur
```

Derleme işlemi biraz zaman alacaktır.

Derleme tamamlandığında Kernel: arch/x86/boot/bzImage is ready (#1) şeklinde bir satır yazmalıdır. Kernelimizin düzgün derlenip derlenmediğini anlamak için aşağıdaki komutu kullanabilirsiniz.

```
file arch/x86/boot/bzImage arch/x86/boot/bzImage: Linux kernel x86 boot executable bzImage,
version 6.6.2 (etapadmin@etahta) #1 SMP PREEMPT_DYNAMIC Sat Nov 25 19:53:19 +03 2023,
RO-rootFS, swap_dev 0XC, Normal VGA
```

Kernel derledikten sonra kendi sistemimizde kullanmak için **/arch/x86/boot/bzImage** konumundan alıp kullanabiliriz. Derlenen kernel'e uygun modüllerin derlenmesi gerekmektedir.

Modul Derleme

Modul ise kernel ile donanım arasında iletişim kuran yazılımlardır. Modüller kernel versiyonuyla aynı versiyonda olmalıdır. Başka versiyon kernel'ler derlenen modül kernel tarafından kullanılamaz. Bundan dolayı kullanılacak modüllerin kernel versiyonuna uygun derlenmesi lazımdır.

```
make modules # tek çekirdekle derleme yapacak yavaş olur
#make modules -j$(proc) #işlemci çekirdek sayısını sistemden alarak yapıyor, hızlı olur
```

Yardımcı Konular

Derleme işlemi biraz zaman alacaktır. Modüller derlendikten sonra sisteme aşağıdaki komutla yüklenir.

Modül Yükleme

Modüller istenilen konuma yüklenebilir. Yükleme konumunu **INSTALL_MOD_PATH** parametresiyle belirleyebiliriz. Eğer belirmezsek /lib/konumuna yüklenecektir.

```
INSTALL_MOD_PATH="$HOME/distro/initrd" make modules_install
```

Strip Yapma

Kernel Yükleme

Header Yükleme

libc Yükleme

busybox Nedir?

Busybox tek bir dosya halinde bulunan birçok araç seçeneğine 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.

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

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.

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

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.

initrd Tasarımı(Debian)

Sistem İçin Gerekli Olan Dosyalar Ve Açılış Süreci

Linux sisteminin açılabilmesi için aşağıdaki 3 dosya yeterli.

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

Bu dosyaları yukarıdaki gibi dizin konumlarına koyduktan sonra, **grub-mkrescue iso/ -o distro.iso #iso dosyamız oluşturulur.** komutuyla **distro.iso** dosyası elde ederiz. Artık iso dosyamız boot edebilen hazırlanmış bir dosyadır. Burada bazı sorulara cevap vermemiz gerekmektedir.

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.

distro/iso/boot/vmlinuz dosyamız kernelimiz oluyor. Ben kullandığım debian sisteminin mevcut kernelini kullandım. İstenirse kernel derlenebilir.

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- **LILLO/GRUB Yazılımı Yükleniyor(grub.cfg dosyası okunuyor ve vmlinuz ve initrd.img devreye giriyor)**
- 4- **vmlinuz initrd.img sistemini belleğe yüklüyor**
- 5- **vmlinuz initrd.img sistemini belleğe yüklüyor**
- 6- **initrd.img içindeki init dosyasındaki işlem sürecine göre sistem işlemlere devam ediyor**
- 7- **initrd.img içindeki init dosyası temel işlemleri ve modülleri yükledikten sonra disk üzerindeki sisteme(/sbin/init) exec switch_root komutuyla süreci devrederek görevini tamamlamış olur**

Yazının devamında sistem için gerekli olan 3 temel dosyanın hazırlanması ve iso yapılıma süreci anlatılacaktır.

initrd Nedir? Nasıl Hazırlanır?

initrd (initial RAM disk), Linux işletim sistemlerinde kullanılan bir geçici dosya sistemidir. Bu dosya sistemi, işletim sistemi açılırken kullanılan bir köprü görevi görür ve gerçek kök dosya sistemine geçiş yapmadan önce gerekli olan modülleri ve dosyaları içerir. Ayrıca, sistem başlatıldığında kök dosya sistemine erişim sağlamadan önce gerekli olan dosyaları yüklemek için de kullanılabilir.

Gerekli olacak dosyalarımızın dizin yapısı ve konumu aşağıdaki gibi olmalıdır. Anlatım buna göre yapılacaktır. Örneğin S1 ifadesi satır 1 anlamında anlatımı kolaylaştırmak için yazılmıştır. Aşağıdaki yapıyı oluşturmak için yapılması gerekenleri adım adım anlatılacaktır.

```
S1- distro/initrd/bin/busybox          #dosya
S2- distro/initrd/bin/kmod             #dosya
S3- distro/initrd/bin/debmod           #dosya
```

```
S4- distro/initrd/bin/insmod          #dosya
S5- distro/initrd/bin/lsmmod         #dosya
S6- distro/initrd/bin/modprobe       #dosya
S7- distro/initrd/bin/rmmod          #dosya
S8- distro/initrd/bin/modinfo        #dosya
S9- distro/initrd/lib/modules/${uname -r}/moduller #dizin
S10- distro/initrd/bin/systemd-udev  #dosya
S11- distro/initrd/bin/udevadm       #dosya
S12- distro/etc/udev/rules.d         #dizin
S13- distro/lib/udev/rules.d         #dizin
S14- distro/initrd/bin/init          #dosya
S15- distro/iso/initrd.img           #dosya
S16- distro/iso/vmlinuz              #dosya
S17- distro/iso/grub/grub.cfg        #dosya
```

Dizin Yapısının oluşturulması

Aşağıdaki komutları çalıştırdığımızda dizin yapımız oluşacaktır.

```
mkdir -p initrd
mkdir -p initrd/bin/
mkdir -p initrd/lib/
ln -s lib initrd/lib64
mkdir -p initrd/lib/modules/
mkdir -p initrd/lib/modules/${uname -r}
mkdir -p initrd/lib/modules/${uname -r}/moduller
mkdir -p initrd/etc/udev/
mkdir -p initrd/lib/udev/
mkdir -p iso
mkdir -p iso/boot
mkdir -p iso/boot/grub
```

S1- distro/initrd/bin/busybox

busybox hakkında bilgi almak için busybox yazısında anlatılmıştır. Burada sisteme nasıl ekleneceği anlatılacaktır. busybox dosyamızın bağımlılıklarının **lddscript.sh** scripti ile initrd içine kopyalayacağız. Yazının devamında **Bağımlılık Tespiti** konu başlığı altında anlatılmıştır.

```
cp /usr/bin/busybox initrd/bin/busybox #sistemden busybox kopyalandı..
lddscript.sh initrd/bin/busybox initrd/ #sistemden busybox bağımlılıkları initrd dizinimize kopyalar.
```

S2-S8 distro/initrd/bin/kmod

kmod yazısında kmod anlatılmıştır. Burada sisteme nasıl ekleneceği anlatılacaktır.

```
cp /usr/bin/kmod initrd/bin/kmod #sistemden kmod kopyalandı..
lddscript.sh initrd/bin/kmod initrd/ #sistemden kmod kütüphaneleri kopyalandı..
ln -s kmod initrd/bin/depmod      #kmod sembolik link yapılarak depmod hazırlandı.
ln -s kmod initrd/bin/insmod      #kmod sembolik link yapılarak insmod hazırlandı.
ln -s kmod initrd/bin/lsmmod      #kmod sembolik link yapılarak lsmod hazırlandı.
ln -s kmod initrd/bin/modinfo     #kmod sembolik link yapılarak modinfo hazırlandı.
ln -s kmod initrd/bin/modprobe    #kmod sembolik link yapılarak modprobe hazırlandı.
ln -s kmod initrd/bin/rmmod       #kmod sembolik link yapılarak rmmod hazırlandı.
```

S9- distro/initrd/lib/modules/\$(uname -r)/moduller

Bu bölümde modüller hazırlanacak. Burada dikkat etmemiz gereken önemli bir nokta kullandığımız kernel versiyonu neyse **initrd/lib/modules/modules** altında oluşacak dizinimiz aynı olmalıdır. Bundan dolayı **initrd/lib/modules/\$(uname -r)** şeklinde dizin oluşturulmuştur. Aşağıda kullandığımız 2. satırdaki **/sbin/depmod --all --basedir=initrd, initrd/lib/modules/\$(uname -r)/moduller** altındaki modüllerimizin indeksinin oluşturuyor.

```
#döngüyle istediğimiz moduller initrd sistemimize dahil ediliyor.
for directory in {crypto,fs,lib} \
    drivers/{block,ata,md,firewire} \
    drivers/{scsi,message,pcmcia,virtio} \
    drivers/usb/{host,storage};
do
    #echo ${directory}
    find /lib/modules/$(uname -r)/kernel/${directory}/ -type f \
        -exec install {} initrd/lib/modules/$(uname -r)/moduller \;
done
/sbin/depmod --all --basedir=initrd    #modüllerin indeks dosyası oluşturuluyor
```

S9- distro/initrd/bin/systemd-udev

udev, Linux çekirdeği tarafından sağlanan bir altyapıdır ve donanım aygıtlarının dinamik olarak algılanmasını ve yönetilmesini sağlar. systemd-udev ise udev'in bir bileşenidir ve donanım olaylarını işlemek için kullanılır. Daha detaylı bilgi için udev yazısında anlatılmıştır. systemd için **/lib/systemd/systemd-udev**, no systemd için **/sbin/udev** kullanılır. Biz systemd için tasarladığımız için **/lib/systemd/systemd-udev** kullanıyoruz.

```
cp /lib/systemd/systemd-udev initrd/bin/systemd-udev #sistemden kopyalandı..
lddscript initrd/bin/systemd-udev initrd/ #sistemden kütüphaneler kopyalandı..
```

S10- distro/initrd/bin/udevadm

udevadm, Linux işletim sistemlerinde kullanılan bir araçtır. Bu araç, udev (Linux çekirdeği tarafından sağlanan bir hizmet) ile etkileşim kurmamızı sağlar. udevadm, sistemdeki aygıtların yönetimini kolaylaştırmak için kullanılır.

udevadm komutu, birçok farklı parametreyle kullanılabilir. İşte bazı yaygın kullanımları:

udevadm info: Bu komut, belirli bir aygıt hakkında ayrıntılı bilgi sağlar. Örneğin, udevadm info -a -n /dev/sda komutunu kullanarak /dev/sda aygıtıyla ilgili ayrıntıları alabilirsiniz.

udevadm monitor:* Bu komut, sistemdeki aygıtlarla ilgili olayları izlemek için kullanılır. Örneğin, udevadm monitor --property komutunu kullanarak aygıtların bağlanma ve çıkarma olaylarını izleyebilirsiniz.

udevadm trigger:* Bu komut, udev kurallarını yeniden değerlendirmek ve aygıtları yeniden tanımak için kullanılır. Örneğin, udevadm trigger --subsystem-match=block komutunu kullanarak blok aygıtlarını yeniden tanımlayabilirsiniz.

udevadm control: Bu komut, udev hizmetini kontrol etmek için kullanılır. Örneğin, udevadm control --reload komutunu kullanarak udev kurallarını yeniden yükleyebilirsiniz.

Bu sadece bazı temel kullanımlardır ve udevadm'nin daha fazla özelliği vardır. Daha fazla bilgi için, man udevadm komutunu kullanarak udevadm'nin man sayfasını inceleyebilirsiniz. **Not:** udevadm systemd ve no systemd için aynı kullanımdadır. İki sistem içinde geçerlidir.

```
cp /bin/udevadm initrd/bin/udevadm #sistemden udevadm kopyalandı..
lddscript initrd/bin/udevadm initrd/ #sistemden kütüphaneler kopyalandı..
```


S12- distro/etc/udev/rules.d--S13- distro/lib/udev/rules.d

"rules" kelimesi, Linux işletim sistemi veya bir programda belirli bir davranışı tanımlayan ve yönlendiren kuralları ifade eder. Bu kurallar, sistem veya programın nasıl çalışacağını belirlemek için kullanılır ve genellikle yapılandırma dosyalarında veya betiklerde tanımlanır.

Linux'ta "rules" terimi, genellikle udev kuralları veya iptables kuralları gibi belirli bileşenlerle ilişkilendirilir.

udev kuralları, Linux çekirdeği tarafından sağlanan bir altyapıdır ve donanım aygıtlarının nasıl tanınacağını ve nasıl işleneceğini belirlemek için kullanılır. Örneğin, bir USB cihazı takıldığında, udev kuralları bu cihazın nasıl adlandırılacağını ve hangi sürücünün kullanılacağını belirleyebilir.

Örnek bir udev kuralı:

```
ACTION=="add", SUBSYSTEM=="usb", ATTR{idVendor}=="1234",  
ATTR{idProduct}=="5678", RUN+="/path/to/script.sh"
```

Bu kural, bir USB cihazı eklendiğinde çalışacak bir betik belirtir. Kural, cihazın üretici kimliği (idVendor) ve ürün kimliği (idProduct) gibi özelliklerini kontrol eder ve belirli bir eylem gerçekleştirir.

Aşağıda sisteme ait kurallar initrd sistemimize kopyalanmaktadır.

```
cp /etc/udev/rules.d -rf initrd/etc/udev/  
cp /lib/udev/rules.d -rf initrd/lib/udev/
```

S14- distro/initrd/bin/init

kernel ilk olarak initrd.img dosyasını ram'e yükleyecek ve ardından **init** dosyasının arayacaktır. Bu dosya bir script dosyası veya binary bir dosya olabilir. Bu tasarımda script dosya olacaktır. İçeriği aşağıdaki gibi olacaktır.

```
cat > initrd/init << EOF  
#!/bin/busybox ash  
PATH=/bin  
/bin/busybox mkdir -p /bin  
/bin/busybox --install -s /bin  
#*****  
export PATH=/bin:/sbin:/usr/bin:/usr/sbin  
  
[ -d /dev ] || mkdir -m 0755 /dev      #/dev dizini yoksa oluştur  
[ -d /root ] || mkdir -m 0700 /root   #/root dizini yoksa oluştur  
[ -d /sys ] || mkdir /sys             #/sys dizini yoksa oluştur  
[ -d /proc ] || mkdir /proc          #/proc dizini yoksa oluştur  
mkdir -p /tmp /run                   # /tmp ve /run dizinleri oluşturuluyor  
  
# sisteme izinler bağlanıyor(yükleniyor)  
mount -t devtmpfs devtmpfs /dev  
mount -t proc proc /proc  
mount -t sysfs sysfs /sys  
mount -t tmpfs tmpfs /tmp  
  
systemd-udevd --daemon --resolve-names=never #modprobe yerine kullanılıyor  
udevadm trigger --type=subsystems --action=add  
udevadm trigger --type=devices --action=add  
udevadm settle || true  
  
mkdir -p /disk                       # /dev/sdal diskini bağlamak için izin oluşturuluyor  
modprobe ext4                        #ext4 modülü yükleniyor harici olarak yüklememiz gerekiyor  
mount /dev/sdal /disk                #diski bağlayalım  
  
# dev sys proc taşıyalım  
mount --move /dev /disk/dev  
mount --move /sys /disk/sys  
mount --move /proc /disk/proc  
  
exec switch_root /disk /sbin/init     #sistemi initrd içindeki initten sdal diskinde olan /sbin/init'e devrediyoruz.  
/bin/busybox ash                     #eğer üst satırdaki görev devir işlemi olmazsa bu satır çalışacak ve tty açılacaktır.  
EOF  
chmod +x initrd/init #init dosyasına çalıştırma izni veriyoruz.  
cd initrd  
find |cpio -H newc -o >initrd.img # initrd.img dosyasını initrd dizinine oluşturacaktır.  
cd ..
```

Yardımcı Konular

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

S15- distro/iso/initrd.img - S16- distro/iso/vmlinuz

initrd.img dosyası kernel(vmlinuz) ile birlikte kullanılan belleğe ilk yüklenen dosyadır. Bu dosyanın görevi sistemin kurulu olduğu diski tanımak için gereken modülleri yüklemek ve sistemi başlatmaktır. Bu dosya /boot/initrd.img-xxx konumunda yer alır. initrd.img dosyası üretmek için

```
cp /boot/vmlinuz-$(uname -r) iso/boot/vmlinuz #sistemde kullandığım kerneli kopyaladım istenirde kernel derlenebilir.
mv initrd/initrd.img iso/boot/initrd.img #daha önce oluşturduğumuz **initrd.img** dosyamızı taşıyoruz.
```

S17- distro/iso/grub/grub.cfg

grub menu dosyası oluşturuluyor.

```
cat > iso/boot/grub/grub.cfg << EOF
linux /boot/vmlinuz
initrd /boot/initrd.img
boot
EOF
```

Yukarıdaki script **iso/boot/grub/grub.cfg** dosyasının içeriği olacak şekilde ayarlanır.

iso Oluşturma

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

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

qemu-system-x86_64 -cdrom distro.iso -m 1G komutuyla çalıştırıp test edebiliriz.

Bağımlılıkların Tespiti

İkili dosyasının iki tür derlenme şekli vardır(statik ve dinamik). Statik derleme gerekli olan kütüphaneleri içerisinde barındıran tek bir dosyadır. Dinamik olan ise gerekli olan kütüphane dosyaları ikili dosya dışında tutulmaktadır. İkili dosyamızın bağımlılıklarının bulunması için aşağıdaki scripti kullanabiliriz. Scripti lddscript.sh dosyası olarak kaydedip kullanabilirsiniz. **bash lddscript.sh /bin/ls /tmp/test** şeklinde kullandığımızda /tmp/test/ dizinine **ls** ikili dosyasının konumunu ve bağımlılıklarını kopyalayacaktır.

```
#!/bin/bash
#bash lddscript binaryPath binaryTarget
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

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

# copy the library dependencies
echo "---> copy libraries"
ldd "${path_to_binary}" | awk -F'[> ]' '{print $(NF-1)}' | while read -r lib
do
    [ -f "$lib" ] && cp -v --parents "$lib" "${target_folder}"
done
```

initrd(bps)

initrd (initial RAM disk), Linux işletim sistemlerinde kullanılan bir geçici dosya sistemidir. Bu dosya sistemi, işletim sistemi açılırken kullanılan bir köprü görevi görür ve gerçek kök dosya sistemine geçiş yapmadan önce gerekli olan modülleri ve dosyaları içerir. Ayrıca, sistem başlatıldığında kök dosya sistemine erişim sağlamadan önce gerekli olan dosyaları yüklemek için de kullanılabilir. Bu bölümü uygulamadan önce uygulam adımlarını daha anlayabilmek için mutlaka initr tasalama konusunu okumalısınız.

Temel Dosyalar

Linux sisteminin açılabilmesi için aşağıdaki 3 dosya yeterlidir. Bu dosyalar yardımıyla sistem açılışı yapılır ve diskimizde bulunan sistemi bu 3 dosya yardımıyla çalıştırır ve hazır hale getiririz. Şimdi sırasıyla 3 dosyamızı nasıl hazırlayacağımızı adım adım uygulayalım. Dosyaları oluşturduktan sonra iso haline getirerek sistemi çalışır hale getirelim.

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

distro/iso/boot/initrd.img dosyası sistemin açılış sürecinde ön işlemleri yaparak gerçek sisteme geçiş sürecini yöneten bir dosyadır. Yazın devamında nasıl hazırlanacağı anlatılacaktır.

distro/iso/boot/vmlinuz dosyamız kernelimiz oluyor. Ben kullandığım debian sisteminin mevcut kernelini kullandım. İstenirse kernel derlenebilir.

distro/iso/boot/grub/grub.cfg dosyamız ise initrd.img ve vmlinuz dosyalarının grub yazılımını nerede bulacağını gösteren yapılandırma dosyasıdır.

linux Açılış Süreci

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**
6. initrd.img içindeki init dosyası temel işlemleri ve modülleri yükledikten sonra disk üzerindeki sisteme(/sbin/init) exec switch_root komutuyla süreci devrederek görevini tamamlamış olur

Yazının devamında sistem için gerekli olan 3 temel dosyanın(initrd.img, vmlinuz, grub.cfg) hazırlanması ve iso yapılma süreci anlatılacaktır.

initrd Dosya İçeriği

initrd.img dosyasını hazırlarken gerekli olacak dosyalarımızın dizin yapısı ve konumu aşağıdaki gibi olmalıdır. Anlatım buna göre yapılacaktır. Örneğin S1 ifadesi satır 1 anlamında anlatımı kolaylaştırmak için yazılmıştır. Aşağıdaki yapıyı oluşturmak için yapılması gerekenleri adım adım anlatılacaktır.

```
S1- $boot/bin/busybox                #dosya
S2- $boot/sbin/kmod                  #dosya
S3- $boot/sbin/debmod                #dosya
S4- $boot/sbin/insmod                #dosya
S5- $boot/bin/lsmmod                 #dosya
S6- $boot/sbin/modprobe              #dosya
S7- $boot/sbin/rmmod                 #dosya
S8- $boot/sbin/modinfo               #dosya
S9- $boot/lib/modules/${uname -r}/modueller #dizin
S10- $boot/bin/udevadm               #dosya
S11- $boot/bin/udev                  #dosya
S12- $boot/etc/udev/rules.d          #dizin
S13- $boot/lib/udev/rules.d          #dizin
S14- $boot/initrd/bin/init            #dosya
S15- distro/iso/initrd.img           #dosya
S16- distro/iso/vmlinuz               #dosya
S17- distro/iso/grub/grub.cfg        #dosya
```

S1-S17 arasındaki dosya ve dizin yapısını hazırladığımız **initrd** adındaki script hazırlayacak ve iso haline getirecektir.

Yardımcı Konular

S1-S17 arasındaki adımları yapacak **initrd** scripti aşağıdaki gibi hazırlandı.

initrd Scripti

```
#!/bin/bash
boot=$HOME/distro/initrd
rm -rf $boot

mkdir -p $HOME/distro
mkdir -p $boot
mkdir -p $boot/bin
#*****hazırlanmış olan bps paketlerimiz yükleniyor*****
./bpsupdate
./bpskur glibc $boot/          # Dağıtımımızın temel kütüphanesini oluşturan paket yükleniyor
./bpskur busybox $boot/        # S1- distro/initrd/bin/busybox paketi yükleniyor
./bpskur kmod $boot/           # S2-S8 distro/initrd/bin/kmod aşamalarını kmod paketi yüklenince oluşur

#*****modul yukleme*****S9- distro/initrd/lib/modules/$(uname -r)/moduller hazırlanıyor
mkdir -p $boot/lib/modules/
mkdir -p $boot/lib/modules/$(uname -r)
mkdir -p $boot/lib/modules/$(uname -r)/moduller
cp /lib/modules/$(uname -r)/kernel/* -prvf $boot/lib/modules/$(uname -r)/moduller/ #sistemden kopyalandı..
/sbin/depmod --all --basedir=$boot #modul indeksi oluşturluyor

./bpskur eudev $boot/          # S10-S13 eudev paketi yüklenerek oluşturur
./bpskur base-file $boot/      # S14- $boot/initrd/bin/init oluşturma
./bpskur util-linux $boot/
./bpskur grub $boot/
./bpskur e2fsprogs $boot/

#*****initrd.img oluşturuluyor*****# S15- distro/iso/initrd.img
cd $boot
find | cpio -H newc -o >../initrd.img
#*****iso *****
mkdir -p $HOME/distro/iso
mkdir -p $HOME/distro/iso/boot
mkdir -p $HOME/distro/iso/boot/grub
mkdir -p $HOME/distro/iso/live || true

#iso dizinine vmlinuz ve initrd.img dosyamız kopyalanıyor
cp /boot/vmlinuz-$(uname -r) $HOME/distro/iso/boot/vmlinuz #sistemde kullandığım kerneli kopyladım istenirde kernel derlenebilir.
mv $HOME/distro/initrd.img $HOME/distro/iso/boot/initrd.img #oluşturduğumuz **initrd.img** dosyamızı taşıyoruz.

#grub menüsü oluşturuluyor..
cat > $HOME/distro/iso/boot/grub/grub.cfg << EOF
linux /boot/vmlinuz net.ifnames=0 biosdevname=0
initrd /boot/initrd.img
boot boot=live
EOF
```

S1- \$boot/bin/busybox

busybox küçük boyutlu dağıtım ve initrd hazırlamada kullanılan, birçok uygulamayı içinde barındıran dosyamızdır. **Temel Paketler** başlığı altında nasıl derleneceği anlatıldı. Derleme ve paket oluşturma aşamalarında **busybox** paketinizi oluşturduğunuzu varsayıyoruz. Burada sisteme nasıl ekleneceği anlatılacaktır.

```
./bpskur busybox $boot/
```

S2-S8 \$boot/bin/kmod

kmod yazısında kmod anlatılmıştır. Burada sisteme nasıl ekleneceği anlatılacaktır. kmod paketi aşağıdaki komut satırıyla kurulmaktadır.

```
./bpskur kmod $boot/
```

Kurulum tamamlandığında paket içerisindeki dosya ve sembolik link dosyaları aşağıdaki gibi **\$boot** konumuna yüklenecektir.

```
$boot/sbin/kmod
ln -s $boot/sbin/kmod $boot/sbin/depmod      #kmod sembolik link yapılarak depmod hazırlandı.
ln -s $boot/sbin/kmod $boot/sbin/insmod      #kmod sembolik link yapılarak insmod hazırlandı.
ln -s $boot/sbin/kmod $boot/bin/lsmode      #kmod sembolik link yapılarak lsmod hazırlandı.
ln -s $boot/sbin/kmod $boot/sbin/modinfo     #kmod sembolik link yapılarak modinfo hazırlandı.
ln -s $boot/sbin/kmod $boot/sbin/modprobe    #kmod sembolik link yapılarak modprobe hazırlandı.
ln -s $boot/sbin/kmod $boot/sbin/rmmod       #kmod sembolik link yapılarak rmmod hazırlandı.
```

S9- \$boot/lib/modules/\$(uname -r)/moduller

Bu bölümde modüller hazırlanacak. Burada dikkat etmemiz gereken önemli bir nokta kullandığımız kernel versiyonu neyse **\$boot/lib/modules/modules** altında oluşacak dizinimiz aynı olmalıdır. Bundan dolayı **\$boot/lib/modules/\$(uname -r)** şeklinde dizin oluşturulmuştur. Aşağıda kullandığımız son satırdaki **/sbin/depmod --all --basedir=initrd, \$boot/lib/modules/\$(uname -r)/moduller** altındaki modüllerimizin indeksini oluşturuyor.

```
mkdir -p $boot/lib/modules/
mkdir -p $boot/lib/modules/$(uname -r)
mkdir -p $boot/lib/modules/$(uname -r)/moduller

cp /lib/modules/$(uname -r)/kernel/* -prvf $boot/lib/modules/$(uname -r)/moduller/ #modüller sistemden kopyalandı..
/sbin/depmod --all --basedir=$boot #modüllerin indeks dosyası oluşturuluyor
```

S10-S13- \$boot/bin/udevadm

udevadm, Linux işletim sistemlerinde kullanılan bir araçtır. Bu araç, udev (Linux çekirdeği tarafından sağlanan bir hizmet) ile etkileşim kurmamızı sağlar. **udevadm** sistemdeki aygıtların yönetimini kolaylaştırmak için kullanılır. **udev** ise udevadm'in bir bileşenidir ve donanım olaylarını işlemek için kullanılır.

```
./bpskur eudev $boot/ # paket kuruluyor
```

Paket kurulunca aşağıdaki gibi bir dizin yapısı ve dosyalar dağıtım dizinimize(\$boot) yüklenecektir.



S14- distro/initrd/bin/init

kernel ilk olarak initrd.img dosyasını ram'e yükleyecek ve ardından **init** dosyasının arayacaktır. Bu dosya bir script dosyası veya binary bir dosya olabilir. **init** ve sistem için gereken temel dosyaları **base-file** paketi olarak hazırladık. **base-file** paketi aşağıdaki komutla kurulur.

```
./bpskur base-files $boot/          # paket kuruluyor
```

*base-file** paketi içindeki **init** script dosyası aşağıdaki gibi hazırlandı.

init Dosyası

```
#!/bin/busybox ash
/bin/busybox mkdir -p /bin
/bin/busybox --install -s /bin
#####
export PATH=/sbin:/bin:/usr/bin:/usr/sbin:

[ -d /dev ] || mkdir -m 0755 /dev
[ -d /root ] || mkdir -m 0700 /root
[ -d /sys ] || mkdir /sys
[ -d /proc ] || mkdir /proc
mkdir -p /tmp /run
touch /dev/null

# devtmpfs does not get automounted for initramfs
mount -t devtmpfs devtmpfs /dev
mount -t proc proc /proc
mount -t sysfs sysfs /sys
mount -t tmpfs tmpfs /tmp
#####init üzerinden dosya script çalıştırmak için****
for x in $(cat /proc/cmdline); do
    case $x in
        init=*)
            init=${x#init=}
            echo " bu bir test :${x#init=}"
            ${x#init=}
            ;;
    esac
done

echo "initrd başlatıldı"
/bin/busybox ash
```

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

S15- distro/iso/initrd.img

initrd.img dosyası kernel(vmlinuz) ile birlikte kullanılan belleğe ilk yüklenen dosyadır. Bu dosyanın görevi sistemin kurulu olduğu diski tanımak için gereken modülleri yüklemek ve sistemi başlatmaktır.

Bu dosya /boot/initrd.img-xxx konumunda yer alır. **\$HOME/distro/initrd.img** konumuna dosyamızı aşağıdaki gibi oluşturulur.

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

initrd.img iso dosyası hazırlamak için **\$HOME/distro/iso/boot/initrd.img** konumuna taşındı.

```
mv $HOME/distro/initrd.img iso/boot/initrd.img # Oluşturulan **initrd.img** dosyası taşınır.
```

S16- distro/iso/vmlinuz

vmlinuz linuxta **kernel** diye ifade edilen dosyadır. Burada kernel derlemek yerine debianda çalışan kernel dosyamı kullandım. Kernel derlediğinizde **vmlinuz** dosyası elde edeceksiniz. Kernel derleme ayrı başlık altında anlatılmaktadır.

```
cp /boot/vmlinuz-$(uname -r) iso/boot/vmlinuz #sistemde kullandığım kerneli kopyaladım istenirde kernel derlenebilir.
```

S17- distro/iso/grub/grub.cfg

grub menu dosyası oluşturuluyor.

```
cat > iso/boot/grub/grub.cfg << EOF
linux /boot/vmlinuz
initrd /boot/initrd.img
boot
EOF
```

Yukarıdaki script **iso/boot/grub/grub.cfg** dosyasının içeriği olacak şekilde ayarlanır.

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

Bu dosyaları yukarıdaki gibi dizin konumlarına koyduktan sonra;

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

Bu komut çalışınca **distro.iso** dosyası elde ederiz. Artık iso dosyamız boot edebilen hazırlanmış bir dosyadır.

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.

ld(linker)

ld (linker) komutu, Linux sistemlerinde derlenmiş nesne dosyalarını birleştirmek ve yürütülebilir dosyalar veya paylaşılan kütüphaneler oluşturmak için kullanılan bir araçtır. Bu komut, derleyicinin çıktısı olan nesne dosyalarını alır ve bunları birleştirerek çalıştırılabilir bir dosya oluşturur.

ld komutu, bir programın bağımlılıklarını çözmek ve gerekli kütüphaneleri eklemek için kullanılır. Bu sayede program, çalıştırıldığında gereken tüm kütüphanelere erişebilir ve doğru şekilde çalışabilir.

Aşağıda basit bir örnek verilmiştir:

```
gcc -c main.c -o main.o
gcc -c helper.c -o helper.o
ld main.o helper.o -o program
```

Bu örnekte, gcc komutuyla main.c ve helper.c dosyaları derlenir ve nesne dosyaları (main.o ve helper.o) oluşturulur. Ardından, ld komutuyla bu nesne dosyaları birleştirilerek program adında bir çalıştırılabilir dosya oluşturulur.

ld komutu, Linux sistemlerinde programların derlenmesi ve çalıştırılması sürecinde önemli bir rol oynar ve programların doğru şekilde çalışmasını sağlar.

Linker Türleri

Linker, bir programın derlenmiş obje dosyalarını birleştirerek çalıştırılabilir bir dosya oluşturan bir yazılımdır. Linker, programın farklı bölümlerini bir araya getirir, sembol tablolarını oluşturur ve bağımlılıkları çözer.

Linux'ta kullanılan yaygın linker çeşitleri şunlardır:

GNU Linker (ld): GNU Projesi'nin bir parçası olan GNU Linker, Linux sistemlerinde en yaygın olarak kullanılan linker'dir. ld, derlenmiş obje dosyalarını birleştirir ve çalıştırılabilir bir dosya oluşturur. Ayrıca, dinamik bağlantı için gerekli olan dinamik kütüphaneleri de yönetir.

Gold Linker: Gold Linker, GNU Linker'a alternatif olarak geliştirilmiş bir linker'dir. Gold Linker, daha hızlı bağlama süreleri sunar ve büyük projelerde performans avantajı sağlar.

LLD (LLVM Linker): LLD, LLVM projesinin bir parçası olan modern bir linker'dir. LLD, hızlı bağlama süreleri ve düşük bellek tüketimi ile bilinir. Ayrıca, birden fazla platformda çalışabilme özelliğine sahiptir.

Bu linker çeşitleri, Linux sistemlerinde programların birleştirilmesi ve çalıştırılabilir dosyaların oluşturulması için kullanılır. Her bir linker'ın kendine özgü avantajları ve kullanım senaryoları vardır. Projenizin gereksinimlerine ve tercihlerinize bağlı olarak uygun olanı seçebilirsiniz.

```
# GNU Linker (ld) kullanım örneği
ld -o program program.o

# Gold Linker kullanım örneği
gold -o program program.o

# LLD (LLVM Linker) kullanım örneği
lld -o program program.o
```

Yardımcı Konular

cmake

cmake ninja ile derleme

```
cd build_folder  
cmake -G Ninja source_folder  
ninja
```

cmake derleme:

```
cd build_folder  
cmake -G Ninja source_folder  
cmake --build .
```

sudoers

sudoers dosyasını düzenleyerek, belirli kullanıcıların veya kullanıcı gruplarının parola sormadan sudo komutlarını çalıştırabilmesini sağlayabilirsiniz. Bunun için aşağıdaki adımları izleyebilirsiniz:

Terminali açın ve sudo visudo komutunu çalıştırın. Bu komut, sudoers dosyasını düzenlemek için kullanılır. Dosya açıldığında, aşağıdaki gibi bir satır bulun:

```
%sudo    ALL=(ALL:ALL) ALL
```

Bu satırın altına, aşağıdaki gibi bir satır ekleyin:

```
deneme ALL=(ALL) NOPASSWD: ALL
```

Burada deneme yerine parola sormadan sudo komutlarını çalıştırmak istediğiniz kullanıcının adını yazın. Dosyayı kaydedin ve kapatın.

Artık belirtilen kullanıcı, parola sormadan sudo komutlarını çalıştırabilecektir. Ancak, bu işlemi yaparken dikkatli olun. Parola sormadan sudo kullanmak, güvenlik risklerine yol açabilir. Bu nedenle, yalnızca güvendiğiniz kullanıcılar için bu yapılandırmayı kullanmanız önerilir.

Bu yöntemle sudoers dosyasını düzenleyerek, belirli kullanıcıların veya kullanıcı gruplarının parola sormadan sudo komutlarını çalıştırabilmesini sağlayabilirsiniz.

Yukarıdaki yöntem yerine aşağıdaki gibi yapabiliriz.

```
sudo touch /etc/sudoers.d/deneme  
sudo chmod 440 /etc/sudoers.d/deneme
```

/etc/sudoers.d/deneme içeriğine aşağıdaki gibi yapılandırılalım.

```
%sudo    ALL=(ALL) NOPASSWD:ALL  
deneme   ALL=(ALL) NOPASSWD:ALL
```

polkit

Polkit, Linux sistemlerinde yetkilendirme ve erişim kontrolü sağlayan bir altyapıdır. Polkit kuralları, belirli kullanıcıların veya kullanıcı gruplarının belirli işlemleri gerçekleştirmesine izin vermek veya engellemek için kullanılır.

Polkit kurallarını eklemek için aşağıdaki adımları izleyebilirsiniz:

Polkit kurallarının bulunduğu dizine gidin. Genellikle **/etc/polkit-1/rules.d/** dizininde bulunur. Bir metin düzenleyici kullanarak yeni bir dosya oluşturun veya mevcut bir dosyayı düzenleyin. Dosya adı, genellikle **XX-name.rules** formatında olmalıdır. Burada XX, kuralların uygulanma sırasını belirten bir sayıdır ve name ise dosyanın açıklamasını temsil eder. Dosyaya aşağıdaki gibi bir polkit kuralı ekleyin:

```
polkit.addRule(function(action, subject) {
    if (action.id == "org.example.customaction" && subject.user == "username") {
        return polkit.Result.YES;
    }
});
```

Yukarıdaki örnekte, **org.example.customaction** adlı bir eylem için **username** kullanıcıasına izin veriliyor. Bu kuralı ihtiyaçlarınıza göre düzenleyebilirsiniz.

Dosyayı kaydedin ve düzenlediğiniz kuralların etkili olması için Polkit servisini yeniden başlatın. Bu işlem için aşağıdaki komutu kullanabilirsiniz:

```
sudo systemctl restart polkit
```

Artık polkit kurallarınız etkinleştirilmiş olmalı ve belirlediğiniz yetkilendirmeler uygulanmalıdır.

Polkit kuralları, sistem yöneticilerinin belirli işlemleri kontrol etmesine ve kullanıcıların erişimini yönetmesine olanak tanır. Bu sayede sistem güvenliği ve yetkilendirme düzeyi daha iyi kontrol edilebilir.

Tüm Uygulamalarda İzin Verme

```
polkit.addRule(function(action, subject) {
    return polkit.Result.YES;
});
```

Bir Gruba İzin Verme

```
polkit.addRule(function(action, subject) {
    if (subject.isInGroup("test")) {
        return polkit.Result.YES;
    }
});
```

Yardımcı Konular

Bir Grup-User-Uygulamaya İzin Verme

```
polkit.addRule(  
  function(action,subject)  
  {  
    if ( (action.id == "org.freedesktop.policykit.exec") &&  
        (action.lookup("user") == "root") &&  
        (action.lookup("program") == "/path/to/script") &&  
        (subject.isInGroup("someGroup") ) )  
      return polkit.Result.YES;  
    return polkit.Result.NOT_HANDLED;  
  }  
);
```

user-dirs

user-dirs, Linux işletim sistemlerinde kullanıcıların özel klasörlerini içeren bir sistemdir. Bu klasörler genellikle "Masaüstü", "Belgeler", "İndirilenler" gibi isimlerle tanımlanır ve kullanıcıların dosyalarını düzenlemelerini kolaylaştırır.

user-dirs klasörünü kaldırmak için aşağıdaki adımları izleyebilirsiniz:

Terminali açın ve aşağıdaki komutu girin:

```
nano ~/.config/user-dirs.dirs
```

Bu komut, user-dirs.dirs dosyasını açacaktır. Bu dosya, kullanıcı klasörlerinin yolunu ve adını içerir.

Dosyayı düzenlemek için, kaldırmak istediğiniz klasörün satırını bulun ve **"#"** işaretiyle başlayan bir yorum satırı yapın. Örneğin, **"Masaüstü"** klasörünü kaldırmak istiyorsanız, satırı aşağıdaki gibi düzenleyin:

```
#XDG_DESKTOP_DIR="$HOME/Masaüstü"
```

Dosyayı kaydetmek ve kapatmak için **"Ctrl + X"** tuşlarına basın, ardından **"Y"** tuşuna basın ve Enter tuşuna basın.

Değişikliklerin etkili olması için oturumu kapatıp tekrar açın veya aşağıdaki komutu girin:

```
xdg-user-dirs-update
```

Bu adımları takip ederek user-dirs klasörünü Linux sisteminizden kaldırabilirsiniz. Ancak, dikkatli olun ve yanlışlıkla önemli dosyaları silmemeye özen gösterin.

Yeni oluşturulacak kullanıcılarda oluşturulmamasını istiyorsak **/etc/skel/.config/user-dirs.dirs** dosyasını silmeliyiz.

Sistem Dili Değiştirme

Dil ayarlama

Sistem dilini ayarlamak için öncelikle /etc/locale.gen dosyamızı aşağıdaki gibi düzenleyelim.

Dil kodlarına /usr/share/i18n/locales içerisinden ulaşabilirsiniz.

Karakter kodlamalara /usr/share/i18n/charmaps içinden ulaşabilirsiniz.

tr_TR.UTF-8 UTF-8

Not: En altta boş bir satır bulunmalıdır.

Ardından /lib64/locale dizini yoksa oluşturalım.

```
mkdir -p /lib64/locale/
```

Şimdi de çevresel değişkenlerimizi ayarlamak için /etc/profile.d/locale.sh dosyamızı düzenleyelim.

```
#!/bin/sh # Language settings export LANG="tr_TR.UTF-8" export LC_ALL="tr_TR.UTF-8"
```

Not: Türkçe büyük küçük harf dönüşümü (i -> İ ve ı -> I) ascii standartına uyumsuz olduğu için LC_ALL kısmını türkçe ayarlamayı önermiyoruz. Bunun yerine C.UTF-8 veya en_US.UTF-8 olarak ayarlayabilirsiniz.

Son olarak locale-gen komutunu çalıştıralım.

locale-gen

Eğer /lib64/locale/ dizine okuma iznimiz yoksa verelim.

```
chmod 755 -R /lib64/locale/
```

1. Yöntem

/etc/default/locale dosyasını root olarak bir metin editörü ile açın.

Türkçe için : LANG=tr_TR.UTF-8 İngilizce için : LANG=en_US.UTF-8

Sistemi yeniden başlattığınızda seçtiğiniz dil aktif olacaktır.

2. Yöntem

/etc/profile.d/locale.sh dosyanı oluşturun içeriğini aşağıdaki gibi ayarlayın.

```
# Language settings export LANG="tr_TR.UTF-8" export LC_ALL="tr_TR.UTF-8"
```

/etc/profile.d/ dizin erişim iznini 755 yapın.

profile

/etc/profile dosya içeriğini aşağıdaki gibi bir betik bulunmalıdır.

/etc/profile dosyanının içerisinde aşağıdaki betik olmalıdır. Bu betik **/etc/profile.d** içerisinde betikler varsa tüm kullanıcılar için çalıştırılmasını sağlar.

```
if [ -d /etc/profile.d ]; then
    for i in /etc/profile.d/*.sh; do
        if [ -r $i ]; then
            . $i
        fi
    done
unset i
```

```
fi
```

3.Yöntem

ayarlarını değiştirmek istediğimiz kullanıcı dizinindeki **~/.profile** dosyasının içeriğine aşağıdaki kod satırını eklemeliyiz.

```
export LANG="tr_TR.UTF-8" export LC_ALL="tr_TR.UTF-8"
```

Kullanıcı Sistem Ayarları

Profile Dosyası

/etc/profile dosyası, Linux sistemlerinde kullanıcıların oturum açtıklarında çalıştırılan bir betik dosyasıdır. Bu dosya, tüm kullanıcılar için ortak kabuk ayarlarını ve ortam değişkenlerini tanımlar. Kullanıcı oturumu başlatıldığında, /etc/profile dosyası sistem genelindeki ayarları yükler ve kullanıcı oturumuna uygular. Bu dosya, kullanıcıların kabuk ortamlarını yapılandırmak ve özelleştirmek için kullanılır. Örneğin, PATH değişkenini tanımlamak veya diğer kabuk ayarlarını yapılandırmak için /etc/profile dosyası düzenlenebilir. Bu dosya, sistem genelinde tutarlı bir kabuk ortamı sağlamak için önemlidir.

```
/etc/profile /etc/profile.d/* /etc/environment ~/.profile, veya ~/.bash_profile veya  
~/.login veya ~/.zprofile giriş kabuğunuza bağlı olarak ~/.pam_environment (yalnızca  
terminalde çalışan kabuklar için) /etc/bash.bashrc, /etc/zshrc, ~/.bashrc, ~/.zshrc,  
vesaire.
```

Not: /etc/profile.d/ dizinine root dışında kullanıcılar erişim sağlaması için 755 yapmalısınız.

**** profile****

/etc/profile dosyasının içerisinde aşağıdaki betik olmalıdır. Bu betik **/etc/profile.d** içerisinde betikler varsa tüm kullanıcılar için çalıştırılmasını sağlar.

```
if [ -d /etc/profile.d ]; then  
  for i in /etc/profile.d/*.sh; do  
    if [ -r $i ]; then  
      . $i  
    fi  
  done unset i  
fi
```

adduser ve useradd Kullanımı

adduser ve useradd komutları, Linux işletim sisteminde kullanıcı hesapları oluşturmak için kullanılan iki farklı komuttur. Bu iki komut arasındaki temel farklar şunlardır:

1. Kullanım Kolaylığı:

adduser, interaktif bir arayüz sağlar ve kullanıcı oluşturma işlemi sırasında bir dizi soru sorarak kullanıcı dostu bir yaklaşım sunar. useradd ise daha düşük seviyeli bir komuttur ve tüm parametreleri elle belirtmenizi gerektirir. Bu nedenle, kullanımı daha teknik ve karmaşıktır.

Örnek kullanım:

language-bash

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

```
# useradd komutuyla kullanıcı oluşturma sudo useradd -m -s /bin/bash yeni_kullanici
```

2. Varsayılan Davranışlar:

adduser, varsayılan olarak kullanıcı için ev dizini oluşturur ve gerekli ayarları yapar. useradd ise temel bir kullanıcı hesabı oluşturur ancak ek parametreler belirtilmediği sürece ek işlemleri gerçekleştirmez.

3. Güvenlik ve Uyumluluk:

adduser, kullanıcı oluştururken bazı güvenlik kontrolleri yapar ve sistem uyumluluğunu sağlamak için ek adımlar atar. useradd ise daha esnek bir yapıya sahiptir ve kullanıcı oluşturma işlemini daha özelleştirilmiş bir şekilde gerçekleştirmenize olanak tanır.

Sonuç olarak, genel olarak adduser komutu, kullanıcı dostu bir arayüz sunar ve standart kullanıcı oluşturma işlemleri için tercih edilirken, useradd komutu daha teknik detaylara hakim olan kullanıcılar tarafından tercih edilebilir. Her iki komut da kullanıcı yönetimi için önemli araçlardır ve ihtiyaca göre tercih edilmelidir.

internet

Linux sistemlerinde internete bağlanmak için genellikle birkaç temel pakete ihtiyaç duyarsınız. Öncelikle, ağ arayüzünüzü yönetmek için net-tools veya iproute2 paketleri gereklidir. Bu paketler, ağ ayarlarınızı yapılandırmanıza ve ağ durumunu kontrol etmenize olanak tanır.

Ayrıca, DHCP üzerinden otomatik IP alımı için dhclient veya dhcpcd gibi DHCP istemcilerine ihtiyacınız olabilir. Eğer statik bir IP yapılandırması yapacaksanız, ifconfig veya ip komutları ile ayarları manuel olarak yapabilirsiniz.

Son olarak, internet bağlantınızı test etmek için ping ve curl gibi araçlar da faydalıdır. Örnek bir DHCP istemcisi kullanarak internete bağlanmak için aşağıdaki komutu kullanabilirsiniz:

```
language-bash
```

```
sudo dhclient eth0
```

Bu komut, eth0 arayüzü üzerinden DHCP sunucusundan IP adresi almanızı sağlar.

Terminal Yönlendirmesi

İlk terminalde :

```
$ tty /dev/pts/0 $ <no need to run any command here, just see the output>
```

İkinci terminalde :

```
$ ls > /dev/pts/0
```

Artık çıktığı ilk terminalde alıyorsunuz

Başka Yol: \$ tty /dev/pts/0

```
$ tty /dev/pts/1
```

Bu TTY'leri varsayarak, birincinin stdout'unu ikinciye yönlendirmek için bunu ilk terminalde çalıştırın:

```
exec 1>/dev/pts/1
```

Not: Artık her komut çıktısı pts/1'de gösterilecektir.

pts/0'ın varsayılan davranış stdout'unu geri yüklemek için:

```
exec 1>/dev/pts/0
```

Gerçek Zamanlı Terminal Yansıtma

terminal 1'de:

```
$ tty /dev/pts/0
```

Yardımcı Konular

terminal 2'de:

```
$ tty /dev/pts/1
```

terminal 2'de:

```
$ exec &> >(tee >(cat >&/dev/pts/0)) ls
```

Çıktı, siz yazarken bile her iki terminalde de gerçek zamanlı olarak gösterilecektir.

OpenRC

Openrc sistem açılışında çalışacak uygulamaları çalıştıran servis yöneticisidir.

Kurulum

Kaynak koddan derlemek için aşağıdaki adımları izlemelisiniz:

```
$ git clone https://github.com/OpenRC/openrc
$ cd openrc
$ meson setup build --prefix=/usr
$ ninja -C build install
```

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.
```

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

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

Servislerin durumunu öğrenmek için **rc-status** komutu kullanılır. Ayrıca sistemdeki servislerin sonraki açılışta hangisinin başlatılacağını öğrenmek için 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
```

Yardımcı Konular

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
}
...
```