

## **IV. HASIL DAN PEMBAHASAN**

### **A. Pengujian Awal Sistem Operasi GX-Linux**

Pengujian awal yang dilakukan secara emulasi memiliki tujuan untuk mengetahui jenis sistem operasi *embedded* yang sudah tertanam dalam modul CSB625, bagaimana sistem tersebut bekerja. Dari pengujian awal ini akan diperoleh data-data yang dapat merepresentasikan bagaimana kemampuan dari sistem tersebut.

Tahapan yang dilakukan dalam pengujian awal, antara lain :

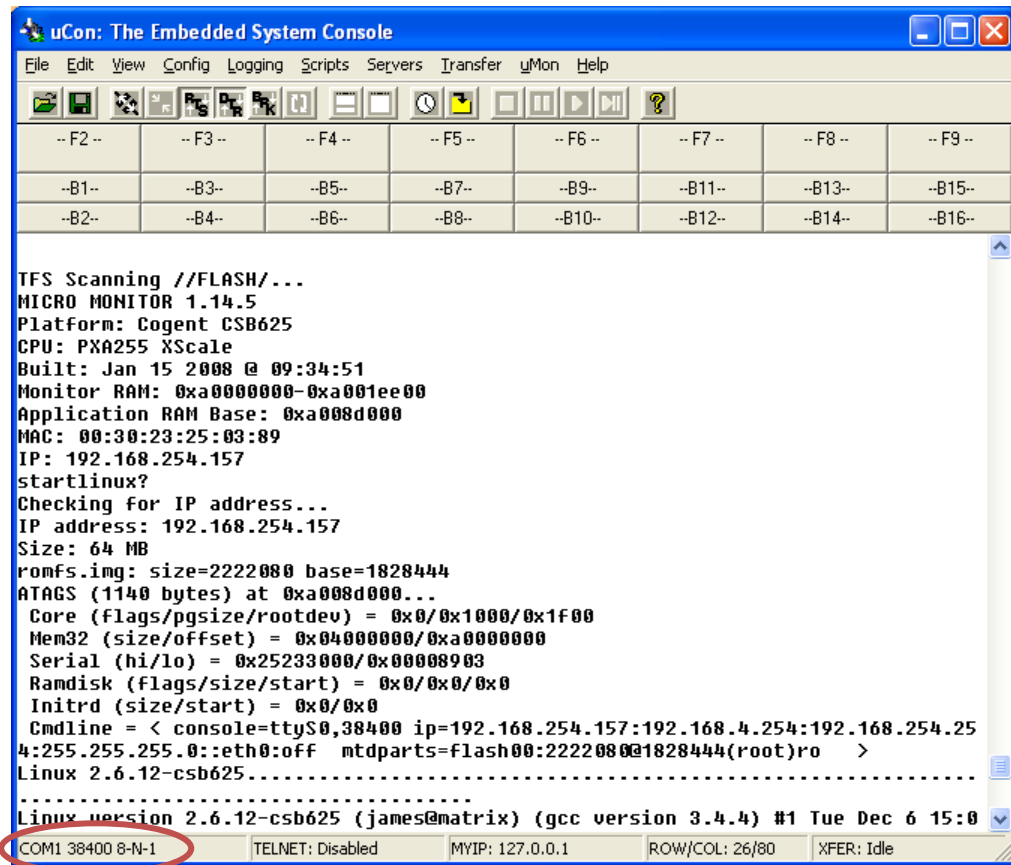
1. Pengujian pengoperasian sistem operasi *embedded* yang ada pada modul CSB625,
2. Pengujian beberapa perangkat *Input/Output*,
3. Pengujian beberapa aplikasi sistem dan pengguna yang telah diimplementasikan didalamnya.

Pengujian dilakukan secara komunikasi serial dengan menggunakan aplikasi Ucon yang didesain khusus untuk pengembangan *firmware* sistem *embedded*.

#### **1. Pengujian Pengoperasian Sistem Operasi GX-Linux**

Komunikasi serial yang digunakan modul CSB625 adalah standar RS-232 dengan konfigurasi baudrate 38400 dan 8-N-1. Setelah melakukan proses

*restarting* pada modul CSB925 maka tampilan jendela Ucon akan berubah seperti gambar 16 berikut.



Gambar 16. Informasi modul CSB625 pada Ucon

Informasi awal yang akan ditampilkan adalah data-data berkaitan dengan platform yang digunakan dengan rincian sebagai berikut.

```
TFS Scanning //FLASH/ ...
MICRO MONITOR 1.14.5
Platform : Cogent CSB625
CPU : PXA255 XScale
Built : Jan 15 2008 @ 09:34:51
Monitor RAM : 0xa0000000-0xa001ee00
Application RAM Base : 0xa008d000
MAC : 00:30:23:25:03:89
IP : 192.168.254.157
UMon>
```

Secara standar setelah proses inialisasi platform selesai dilakukan, maka sistem akan masuk ke dalam *bootloader* Micromonitor. Dari *bootloader* inilah kita bisa melakukan konfigurasi sistem lebih lanjut sehingga sistem dapat berjalan secara otomatis ketika mendapatkan pasokan daya listrik.

Berikut ini adalah proses inialisasi dan *querying user* hingga masuk ke dalam sistem linux ketika *file* `startlinux` dieksekusi secara manual.

```
uMON>startlinux
Checking for IP address...
IP address: 192.168.254.157
Size: 64 MB
romfs.img: size=2222080 base=1828444
ATAGS (1140 bytes) at 0xa008d000...
Core (flags/pgsize/rootdev) = 0x0/0x1000/0x1f00
Mem32 (size/offset) = 0x04000000/0xa0000000
Serial (hi/lo) = 0x25233000/0x00008903
Randisk (flags/size/start) = 0x0/0x0/0x0
Initrd (size/start) = 0x0/0x0

0x001be65c-0x003dce5c : "root"
TCP established hash table entries: 4096 (order: 3, 32768 bytes)
TCP bind hash table entries: 4096 (order: 2, 16384 bytes)
cirrus start: requesting interrupt 37

=====
#
#      @@@@@@ @ @ @ @ *
#      @ @ @ @ @ @ @ @ @ @ (TM)
#      @ @ @ @ @ @ @ @ @ @ @ @ @ @
#      @ @ @ @ @ @ @ @ @ @ @ @ @ @
#      @ @ @ @ @ @ @ @ @ @ @ @ @ @
#      @ @ @ @ @ @ @ @ @ @ @ @ @ @
#      @ @ @ @ @ @ @ @ @ @ @ @ @ @
#
#=====
#
#      GX-Linux(tm) - Standard and Professional Platforms
#      (c) 2004-2005 Microcross, Inc.
#=====

BusyBox v1.00 (2005.08.02-15:56+0000) Built-in shell (ash)
Enter 'help' for a list of built-in commands.

#
```

Untuk menjadikan sistem operasi GX-Linux dapat berjalan secara otomatis ketika target dinyalakan, maka *flag* dari *file* `startlinux` harus dikonfigurasi menjadi bersifat *bootable-executable* dengan mengetikkan perintah berikut pada *Umon console*.

```
UMon> tfs -feB cp startlinux startlinux
```

Dengan mengubah *flag startlinux* menjadi *bootable-executable*, ketika target dinyalakan maka secara otomatis *file* tersebut akan dijalankan pertama kali pada saat *boot system* untuk melakukan inisialisasi sistem dan *querying user* untuk masuk ke sistem operasi GX-Linux. *User* yang digunakan untuk masuk ke dalam sistem operasi adalah *root* dan tanpa disertai dengan kata sandi.

```
TFS Scanning //FLASH/...
MICRO MONITOR 1.14.5
Platform: Cogent CSB625
CPU: PXA255 XScale
Built: Jan 15 2008 @ 09:34:51
Monitor RAM: 0xa0000000-0xa001ee00
Application RAM Base: 0xa008d000
MAC: 00:30:23:25:03:89
IP: 192.168.254.157
startlinux?
Checking for IP address...
IP address: 192.168.254.157
Size: 64 MB
romfs.img: size=2222080 base=1828444
ATAGS (1140 bytes) at 0xa008d000...
Core (flags/pgsize/rootdev) = 0x0/0x1000/0x1f00
Mem32 (size/offset) = 0x04000000/0xa0000000
Serial (hi/lo) = 0x25233000/0x00008903
Ramdisk (flags/size/start) = 0x0/0x0/0x0
Initrd (size/start) = 0x0/0x0

Creating 1 MTD partitions on "flash00":
0x001be65c-0x003dce5c : "root"
TCP established hash table entries: 4096 (order: 3, 32768 bytes)
TCP bind hash table entries: 4096 (order: 2, 16384 bytes)
cirrus_start: requesting interrupt 37
```

```
=====
#
#  aaaaaa  a  a      a      *
#  a      a  a      a      (TM)
#  a      a  a      a      a  a  a  a  a  a
#  a  aaaa  a      aaaa  a      a  a  a  a  a  a
#  a  a      a  a      a      a  a  a  a  a  a
#  a  a      a  a      a      a  a  a  a  a  a
#  aaaaaa  a      a      aaaaaa  a  a  a  aaaaa  a  a
#
#=====
#
#  GX-Linux(tm) - Standard and Professional Platforms
#  (c) 2004-2005 Microcross, Inc.
#=====

BusyBox v1.00 (2005.08.02-15:56+0000) Built-in shell (ash)
Enter 'help' for a list of built-in commands.

# █
```

Berikut ini adalah tampilan jendela Ucon pada saat proses *restart* dan *shutdown* target. Proses restarting dilakukan dengan mengetikkan perintah *reboot* sedangkan untuk *shutdown* perintah yang digunakan adalah *poweroff*.

Setelah proses *shutdown*, indikator *power* serta proses yang ada pada *board* tidak langsung padam, akan tetapi hal ini tidak menimbulkan masalah jika kita melepaskan sumber daya listrik dari target.

```
# reboot
The system is going down NOW !!
Sending SIGTERM to all processes.
Sending SIGKILL to all processes.
enable_irq(2) unbalanced from c014f80c
Please stand by while rebooting the system.
Restarting system.
TFS Scanning //FLASH/...
MICRO MONITOR 1.14.5
Platform: Cogent CSB625
CPU: PXA255 XScale
Built: Jan 15 2008 @ 09:34:51
Monitor RAM: 0xa0000000-0xa001ee00
Application RAM Base: 0xa008d000
MAC: 00:30:23:25:03:89
IP: 192.168.254.157
startlinux?
█

# poweroff
The system is going down NOW !!
Sending SIGTERM to all processes.
Sending SIGKILL to all processes.
enable_irq(2) unbalanced from c014f80c
The system is halted. Press Reset or turn off power
Shutdown: hda
Power down.
```

Kedua perintah tersebut jika dilihat dalam struktur *root filesystem* berada dalam direktori */sbin*.

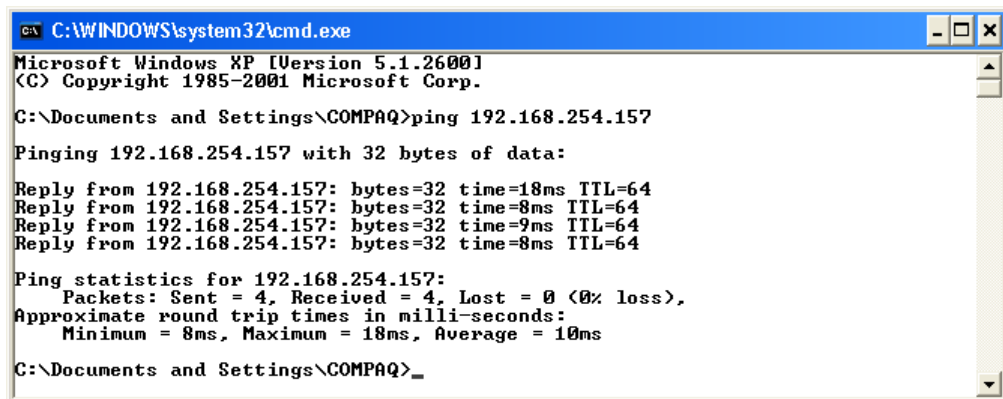
```
# cd sbin/
# ls
halt          insmod        modprobe      reboot        swapoff
ifconfig      klogd         pivot_root    rmmod         swapon
init          lsmod         poweroff      route         syslogd
#
```

## 2. Pengujian Beberapa Perangkat Input/output Modul CSB625

Dari informasi yang diperoleh pada saat inisialisasi *platform* pada aplikasi Ucon, dapat diketahui alamat IP dari target yang sudah dikonfigurasi sebelumnya sehingga dapat dengan mudah dilakukan pengujian perangkat jaringan ini. Alamat IP target adalah 192.168.254.157 dengan *subnet mask* 255.255.255.0. Penulis mengkonfigurasi alamat IP *host* sesuai dengan

kelas dan *network* yang ada pada target yakni 192.168.254.100 dengan *subnet mask* 255.255.255.0.

Hasil uji coba koneksi antara *host* dengan target pada sistem operasi Windows ditunjukkan pada gambar 17 berikut.



```

C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\COMPAQ>ping 192.168.254.157

Pinging 192.168.254.157 with 32 bytes of data:

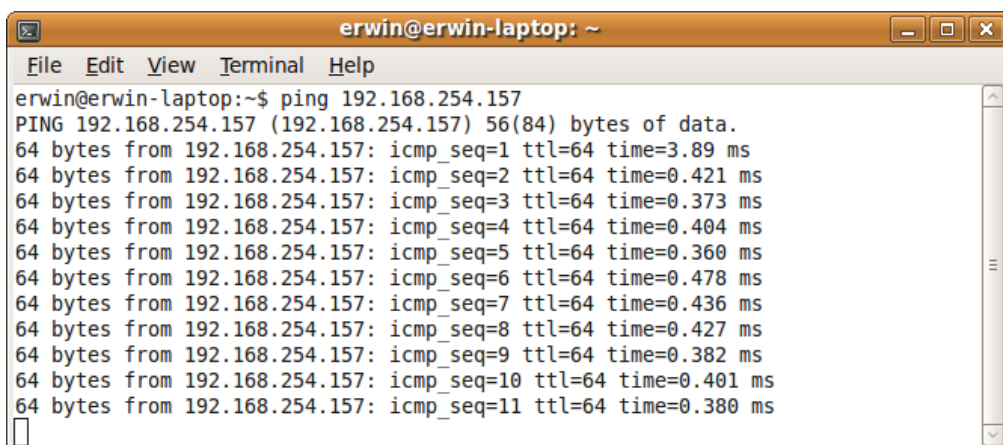
Reply from 192.168.254.157: bytes=32 time=18ms TTL=64
Reply from 192.168.254.157: bytes=32 time=8ms TTL=64
Reply from 192.168.254.157: bytes=32 time=9ms TTL=64
Reply from 192.168.254.157: bytes=32 time=8ms TTL=64

Ping statistics for 192.168.254.157:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 8ms, Maximum = 18ms, Average = 10ms

C:\Documents and Settings\COMPAQ>_
  
```

Gambar 17. Uji koneksi *ethernet* pada windows

Hasil uji coba koneksi antara *host* dengan target pada sistem operasi Linux ditunjukkan pada gambar 18 berikut.



```

erwin@erwin-laptop: ~
File Edit View Terminal Help
erwin@erwin-laptop:~$ ping 192.168.254.157
PING 192.168.254.157 (192.168.254.157) 56(84) bytes of data.
64 bytes from 192.168.254.157: icmp_seq=1 ttl=64 time=3.89 ms
64 bytes from 192.168.254.157: icmp_seq=2 ttl=64 time=0.421 ms
64 bytes from 192.168.254.157: icmp_seq=3 ttl=64 time=0.373 ms
64 bytes from 192.168.254.157: icmp_seq=4 ttl=64 time=0.404 ms
64 bytes from 192.168.254.157: icmp_seq=5 ttl=64 time=0.360 ms
64 bytes from 192.168.254.157: icmp_seq=6 ttl=64 time=0.478 ms
64 bytes from 192.168.254.157: icmp_seq=7 ttl=64 time=0.436 ms
64 bytes from 192.168.254.157: icmp_seq=8 ttl=64 time=0.427 ms
64 bytes from 192.168.254.157: icmp_seq=9 ttl=64 time=0.382 ms
64 bytes from 192.168.254.157: icmp_seq=10 ttl=64 time=0.401 ms
64 bytes from 192.168.254.157: icmp_seq=11 ttl=64 time=0.380 ms
  
```

Gambar 18. Uji koneksi *ethernet* pada Linux

Dari hasil uji coba tersebut dapat disimpulkan bahwa sistem telah dapat melakukan *reply* paket data yang dikirimkan, sehingga perangkat *ethernet* yang ada dapat digunakan untuk komunikasi. Waktu yang digunakan dalam proses pengiriman dan penerimaan paket data juga relatif stabil antara 8 - 9 ms pada Windows dan 0.3 – 0.5 ms pada Linux. Dari data ini dapat disimpulkan bahwa kondisi konektivitas *ethernet* pada sistem *embedded CSB625* cukup stabil untuk melakukan komunikasi data dalam jaringan.

Pengujian perangkat *input/output* selanjutnya adalah *port* USB. Dalam pengujian ini penulis menggunakan USB *flashdisk* dengan kapasitas 2GB. Dari hasil pengujian yang dilakukan, ketika USB *flashdisk* dipasangkan pada *port* USB yang ada maka secara otomatis akan dideteksi oleh sistem dengan informasi yang sesuai dengan spesifikasi *device* yang dipasangkan seperti berikut.

```
# Vendor: SanDisk Model: Cruzer Micro Rev: 8.01
Type: Direct-Access ANSI SCSI revision: 00
SCSI device sda: 3940479 512-byte hdwr sectors (2018 MB)
sda: Write Protect is off
sda: assuming drive cache: write through
SCSI device sda: 3940479 512-byte hdwr sectors (2018 MB)
sda: Write Protect is off
sda: assuming drive cache: write through
Attached scsi removable disk sda at scsi0, channel 0, id 0, lun 0
```

Device yang dipasangkan berada dalam struktur direktori `/dev` yaitu `/dev/sda1`. Penulis melakukan proses *mounting device* tersebut dengan mengarahkannya ke direktori `/mnt/cf` dengan mengetikkan perintah

```
# mount /dev/sda1 /mnt/cf/
```

Ketika pointer *console* diarahkan ke direktori `cf` dan dimasukkan perintah `ls` maka isi dari USB *flashdisk* tersebut akan dapat dibaca seperti berikut.

```
# cd cf/
# ls
2000ju~1.doc  csbtes~1.doc  krikrc~1      shahih~1.pdf  tutorial
acroni~1.rar  driver~1.txt  lab           smadav80.zip  ucon_i~1.zip
amela         erwina~1.jpg  lapora~1      software       umon
arm           formad~1.doc  layout~1      spinrite.rar   esmad~1
autorun.inf   hplase~1.exe  pramod~1.zip  suratt~1.doc
br            htpe7.exe     privacy       testdi~1.2
buildr~1      islam         rabint~1.xls  tips.doc
coding.doc    keagen~1.zip  rahasi~1.doc  trikin~1.doc
```

Untuk melihat *device* apa saja yang telah dilakukan proses *mount* pada sistem dapat dilakukan dengan memasukkan perintah `df`.

```
# df
Filesystem          1k-blocks    Used Available Use% Mounted on
/dev/mtdblock0        2170         2170         0 100% /
/dev/shm             16384          0      16384    0% /tmp
/dev/shm             16384          4      16380    0% /var
/dev/sda1           1964072    1561608    402464    80% /mnt/cf
```

Pada informasi diatas, dapat dilihat adanya *device* baru yakni `/dev/sda1` (USB *flashdisk*) yang di-*mounting* pada direktori `/mnt/cf`.

Sedangkan untuk proses *unmount* yang merupakan proses untuk melepaskan sistem berkas tertentu dari direktori utama, cukup dengan memasukkan perintah seperti berikut.

```
# umount /dev/sda1
```

Maka pada daftar *device* yang telah di-*mounting* akan berubah menjadi seperti berikut.

```
# df
Filesystem          1k-blocks    Used Available Use% Mounted on
/dev/mtdblock0        2170         2170         0 100% /
/dev/shm             16384          0      16384    0% /tmp
/dev/shm             16384          4      16380    0% /var
```

### 3. Pengujian Beberapa Aplikasi Sistem dan Pengguna

Beberapa aplikasi standar telah dimasukkan ke dalam sistem operasi GX-Linux, yang ditempatkan pada direktori `/bin` pada *root filesystem* seperti



aplikasi untuk melakukan *mounting device*. Berikut ini adalah struktur lengkap direktori `/bin` dengan aplikasi-aplikasinya.

```
# cd bin/
# ls
ash      cp        false    kill      mount     rmdir     touch
busybox  date      fgrep    ln         mv         sed        true
cat      dd        grep     ls         ping       sh         umount
chgrp    df        gunzip   mkdir      ps         sleep      uname
chmod    dmesg     gzip     mknod     pwd        sync       vi
chown    echo      hostname more       rm         tar        zcat
```

Pengujian dilakukan pada beberapa aplikasi yang ada untuk mengetahui apakah aplikasi tersebut dapat berjalan dengan baik atau tidak pada target.

a. *Ls*

Merupakan *script* yang digunakan untuk melihat isi dari sebuah direktori.

Dibawah ini menunjukkan isi dari direktori utama *root filesystem ( / )* linux yang ada pada target.

```
# ls
bin      lib      sbin     var
dev      mnt      tmp      var-image.tar.gz
etc      proc     usr
```

b. *Ping*

Perintah ini digunakan untuk melakukan uji koneksi *ethernet* dengan mengirimkan paket data dan menunggu *reply* dari tujuan. Berikut adalah hasil pengujian perintah *ping* dari sistem target dengan tujuan PC *host*.

```
# ping 192.168.254.100
PING 192.168.254.100 (192.168.254.100): 56 data bytes
64 bytes from 192.168.254.100: icmp_seq=0 ttl=128 time=1.1 ms
64 bytes from 192.168.254.100: icmp_seq=1 ttl=128 time=0.6 ms
64 bytes from 192.168.254.100: icmp_seq=2 ttl=128 time=0.5 ms
64 bytes from 192.168.254.100: icmp_seq=3 ttl=128 time=0.6 ms
64 bytes from 192.168.254.100: icmp_seq=4 ttl=128 time=0.5 ms
```

IP 192.168.254.100 adalah alamat IP yang digunakan oleh PC *host*.

### c. *Cp*

Merupakan *script* yang digunakan untuk proses penyalinan data baik antar direktori maupun antar *device*. Pada pengujian dilakukan proses penyalinan data dari USB *flashdisk* ke direktori sistem pada target dan sebaliknya. Berikut ini adalah hasil pengujian proses *copy* data.

```
# cp -r /var/www/ /mnt/cf/
# ls
2000ju~1.doc  csbtes~1.doc  krikrc~1      shahih~1.pdf  tutorial
acroni~1.rar  driver~1.txt  lab           smadav80.zip  ucon_i~1.zip
amela         erwina~1.jpg  lapora~1      software       umon
arm           formad~1.doc  layout~1      spinrite.rar   www
autorun.inf   hplase~1.exe  pramod~1.zip  suratt~1.doc   äsmad-~1
br            htpe7.exe     privacy       testdi~1.2
buildr~1      islam         rabint~1.xls  tips.doc
coding.doc    keagen~1.zip  rahasi~1.doc  trikin~1.doc
# cp /mnt/cf/tips.doc /var/
# cd
# cd var/
# ls
cache      log        run        spool      tips.doc   www
```

Direktori *www* target yang berada pada */var* disalin ke USB *flashdisk* dan *file* *tips.doc* yang berada pada USB *flashdisk* disalin ke direktori */var* target. Proses *copying file* ini juga merepresentasikan transfer data yang terjadi antara *host* dengan target. Dari data-data di atas dapat dilihat bahwa proses transfer data antara *device* ke sistem dan sebaliknya sistem ke *device* tidak mengalami masalah. Hal ini menjadi sebuah kesimpulan bahwa *port USB* pada sistem *embedded CSB625* cukup baik untuk melakukan transfer data.

Berdasarkan pengujian yang telah dilakukan terhadap sistem operasi *embedded GX-Linux* pada modul *CSB625*, perangkat *input/output* dan beberapa aplikasi, dapat diketahui bahwa sistem tersebut telah mampu melakukan fungsi-fungsi seperti layaknya sistem operasi pada *workstation* secara minimal karena terbatasnya sumber daya *hardware* dan aplikasi yang telah diimplementasikan.

## **B. Implementasi Sistem Operasi Baru Berbasis *Open Source Linux***

Ada dua jenis metode yang dapat digunakan dalam implementasi *embedded linux system* yaitu secara manual dan secara otomatis dengan menggunakan *tools* yang sudah dibuat oleh para *engineer* yang telah banyak berkecimpung dalam dunia *embedded system* seperti *crosstool* dan *buildroot*. Metode implementasi secara otomatis memiliki keunggulan jika dibandingkan secara manual. Jika menggunakan metode manual, maka kita harus memiliki referensi yang kuat terutama berkaitan dengan versi aplikasi-aplikasi yang akan digunakan sehingga munculnya *error* dalam proses implementasi bisa diminimalisir. Proses konfigurasi dan instalasi dilakukan tahap demi tahap. Akan tetapi, jika menggunakan metode otomatis maka proses konfigurasi dan instalasi akan dilakukan secara langsung oleh sistem dengan pengaturan yang telah ditentukan oleh *engineer* pembuatnya. Proses yang dilakukan juga menjadi lebih singkat karena hanya dengan menggunakan beberapa baris perintah saja. Hal ini juga akan sangat membantu terutama bagi para pemula dalam perancangan *embedded linux system*.

Dalam penelitian ini, penulis menerapkan kedua metode implementasi yang telah dijelaskan sebelumnya dalam melakukan konfigurasi dan instalasi paket-paket aplikasi yang digunakan dalam implementasi *embedded linux* untuk target.

### **1. Metode Implementasi Secara Manual**

Langkah awal dalam tahapan implementasi *embedded linux system* adalah melakukan konfigurasi dan instalasi *Cross-platform development toolchain*.

*Toolchain* merupakan *cross-develop applications* yang digunakan untuk mengembangkan sistem operasi yang akan dijalankan pada target untuk berbagai jenis arsitektur yang meliputi beberapa *software*, antara lain : *binary binutils*, *C compiler* dan *C library*. Prosedur dalam instalasi *toolchain* meliputi lima tahapan utama, yaitu :

- a. Instalasi *kernel header*,
- b. Instalasi *binary utilities*,
- c. Instalasi *bootstrap compiler*,
- d. Instalasi *C library*,
- e. Instalasi *full compiler*.

Sebelum melakukan konfigurasi dan instalasi *toolchain*, pertama kali yang harus dilakukan adalah membuat direktori untuk *project* yang akan dikembangkan. Beberapa direktori yang harus dibuat seperti ditunjukkan dalam tabel 2 berikut ini.

Tabel 2. *Layout direktori project*

| Direktori          | Isi   |
|--------------------|---|
| <i>Bootldr</i>     | <i>Bootloader</i> untuk target  |
| <i>Build-tools</i> | Paket aplikasi dan direktori yang dibutuhkan untuk <i>cross-platform development toolchain</i>            |
| <i>Debug</i>       | <i>Debugging tools</i> dan paket aplikasi yang berkaitan  |
| <i>Doc</i>         | Semua dokumentasi yang diperlukan <i>project</i>  |
| <i>Images</i>      | <i>Binary image bootloader</i> , <i>kernel</i> dan <i>root filesystem</i> yang siap digunakan oleh target |
| <i>Kernel</i>      | Versi <i>kernel</i> yang berbeda untuk target   |
| <i>Project</i>     | <i>Source code</i> yang dideskripsikan untuk <i>project</i>   |
| <i>Rootfs</i>      | <i>Root filesystem</i> yang akan dijalankan oleh target   |
| <i>Sysapps</i>     | Aplikasi-aplikasi yang diperlukan oleh target   |
| <i>Tmp</i>         | Direktori sementara untuk eksperimen dan menyimpan file <i>transient</i>                                  |
| <i>tools</i>       | <i>Cross-platform development toolchain</i> yang lengkap dan <i>library C</i>                             |

Penulis menempatkan direktori-direktori *project* tersebut dalam sebuah direktori utama dengan nama *ikhwan-project* pada *home directory*. Jika dilihat melalui *console* dengan mengetikkan perintah

```
erwin@erwin-laptop:~$ ls -l ~/ikhwan-project
```

maka akan ditampilkan informasi sebagai berikut.

```
total 44

drwxr-xr-x 2 erwin erwin 4096 2010-01-01 11:19 bootldr
drwxr-xr-x 2 erwin erwin 4096 2010-01-01 11:24 build-tools
drwxr-xr-x 2 erwin erwin 4096 2010-01-01 11:22 debug
drwxr-xr-x 2 erwin erwin 4096 2010-01-01 11:22 doc
drwxr-xr-x 2 erwin erwin 4096 2010-01-01 11:22 images
drwxr-xr-x 2 erwin erwin 4096 2010-01-01 11:24 kernel
drwxr-xr-x 2 erwin erwin 4096 2010-01-01 11:22 project
drwxr-xr-x 2 erwin erwin 4096 2010-01-01 11:22 rootfs
drwxr-xr-x 2 erwin erwin 4096 2010-01-01 11:22 sysapps
drwxr-xr-x 2 erwin erwin 4096 2010-01-01 11:23 tmp
drwxr-xr-x 2 erwin erwin 4096 2010-01-01 11:23 tools
```

Selanjutnya ruang kerja yang akan digunakan untuk konfigurasi dan instalasi sebaiknya diatur terlebih dahulu untuk memudahkan proses yang akan dilakukan. Pengaturan ruang kerja dilakukan dengan mengetikkan beberapa perintah berikut dalam *console*

```
erwin@erwin-laptop:~ $ export PROJECT=ikhwan-project
erwin@erwin-laptop:~ $ export PRJROOT=/home/erwin/${PROJECT}
erwin@erwin-laptop:~ $ export TARGET=arm-linux
erwin@erwin-laptop:~ $ export PREFIX=${PRJROOT}/tools
erwin@erwin-laptop:~ $ export
TARGET_PREFIX=${PREFIX}/${TARGET}
erwin@erwin-laptop:~ $ export PATH=${PREFIX}/bin:${PATH}
```

*Value* dari variabel *TARGET* disesuaikan dengan jenis arsitektur target yang akan dikembangkan. Pada penelitian ini target yang akan dikembangkan memiliki jenis arsitektur ARM sehingga *value* dari variabel target adalah *arm-linux*. Dengan pengaturan ruang kerja seperti diatas maka untuk masuk ke direktori *project* cukup dengan mengetikkan perintah.

```
erwin@erwin-laptop:~ $ cd $PRJROOT
```

Sehingga pointer pada *console* akan menunjukkan direktori *project* yang sudah diatur sebelumnya.

```
erwin@erwin-laptop:~/ikhwan-project$
```

Mengacu pada tabel 2, maka *toolchain* akan diinstal pada direktori *build-tools* sehingga perlu dilakukan persiapan terlebih dahulu pada direktori tersebut berupa pembuatan beberapa direktori baru untuk *toolchain*.

```
erwin@erwin-laptop:~/ikhwan-project$ cd ${PRJROOT}/build-
tools

erwin@erwin-laptop:~/ikhwan-project/build-tools$ mkdir
build-binutils build-boot-gcc build-glibc build-gcc
```

Paket-paket *software* yang akan diinstal untuk *toolchain* juga ditempatkan pada direktori *build-tools*, sehingga jika dilihat melalui *console* dengan perintah

```
erwin@erwin-laptop:~/ikhwan-project/build-tools$ ls -l
```

maka akan ditampilkan informasi berikut

```
total 42788

-rwxrwxrwx 1 erwin erwin 16378360 2009-12-25 08:36 binutils-
2.16.1.tar.gz

drwxr-xr-x 2 erwin erwin      4096 2010-01-01 11:30 build-
binutils

drwxr-xr-x 2 erwin erwin      4096 2010-01-01 11:30 build-
boot-gcc

drwxr-xr-x 2 erwin erwin      4096 2010-01-01 11:30 build-gcc

drwxr-xr-x 2 erwin erwin      4096 2010-01-01 11:30 build-
glibc
-rwxrwxrwx 1 erwin erwin 12911721 2009-12-19 11:18 gcc-
2.95.3.tar.gz

-rwxrwxrwx 1 erwin erwin 14441250 2009-12-25 08:46 glibc-
2.2.2.tar.gz
```

### a. Instalasi *Kernel Headers*

Instalasi *kernel header* merupakan langkah awal dalam melakukan instalasi *toolchain* karena akan digunakan dalam instalasi aplikasi *toolchain* yang lainnya. Kernel yang digunakan dalam penelitian ini adalah kernel linux versi 2.6.11.3. *File* kernel linux ditempatkan pada direktori *kernel* yang telah dibuat sebelumnya. Selanjutnya kita melakukan ekstraksi *file* kernel tersebut dengan mengetikkan perintah.

```
erwin@erwin-laptop:~/ikhwan-project/kernel$ tar xvfz
linux-2.6.11.3.tar.gz
```

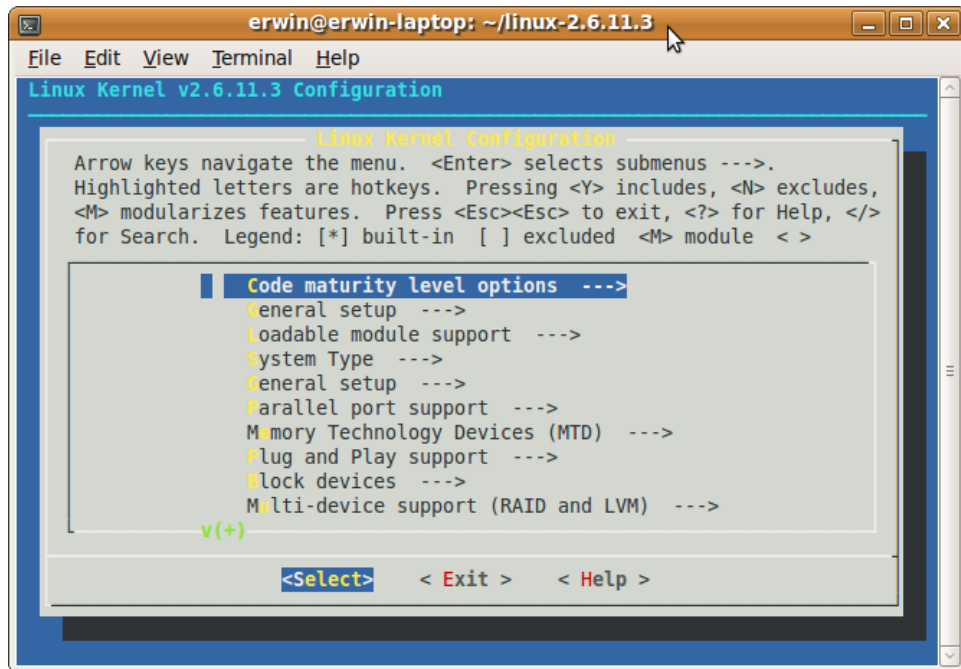
proses ekstraksi kernel akan menghasilkan *file-file* kernel linux yang berada dalam sebuah direktori dengan nama yang sesuai dengan versi kernel yang digunakan yaitu linux-2.6.11.3. Setelah kernel diekstraksi maka selanjutnya dapat dilakukan proses konfigurasi kernel untuk menentukan fitur apa saja yang akan digunakan pada *target system*.

Konfigurasi kernel dilakukan dengan mengetikkan perintah berikut.

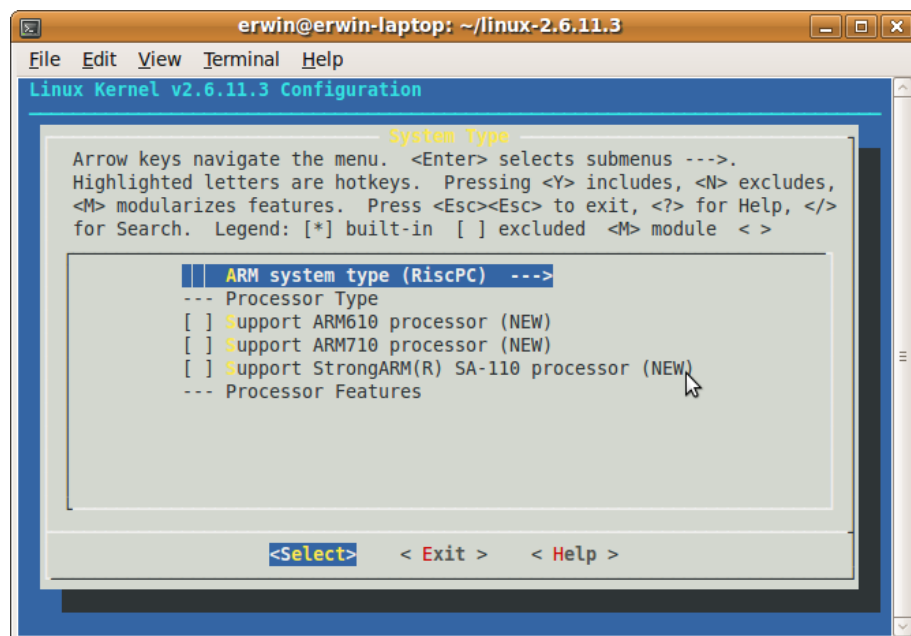
```
erwin@erwin-laptop:~/ikhwan-project/kernel$ cd linux-
2.6.11.3/

erwin@erwin-laptop:~/ikhwan-project/kernel/linux-
2.6.11.3$ make ARCH=arm CROSS_COMPILE=i386-linux-
menuconfig
```

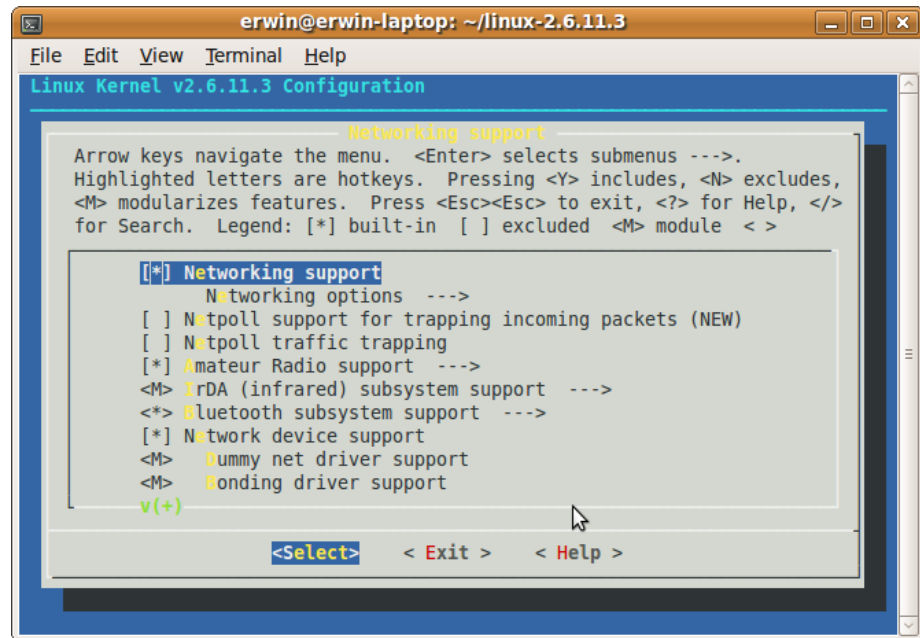
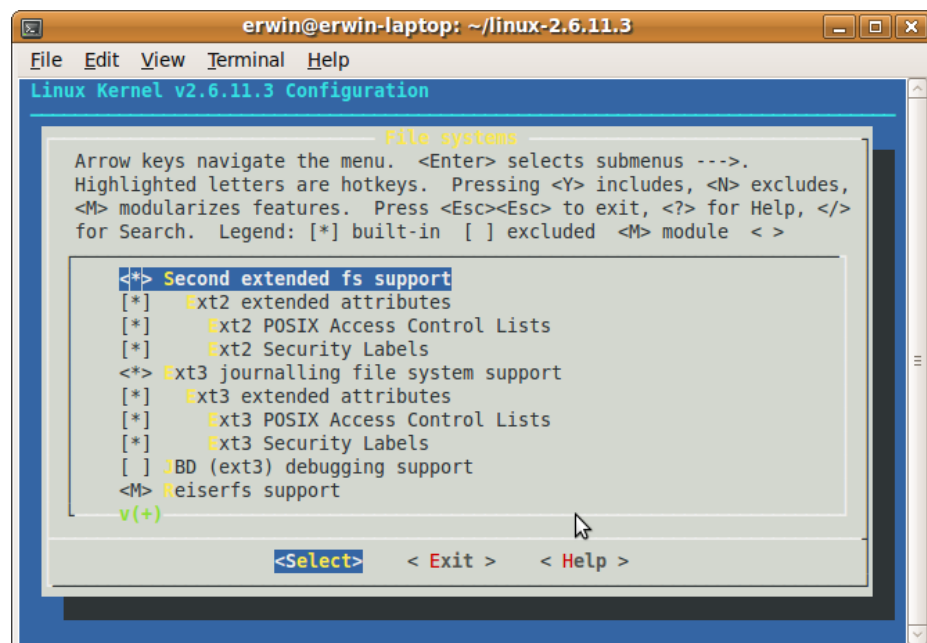
*Value* variabel ARCH merupakan jenis arsitektur *target system*, sedangkan *value* variabel CROSS\_COMPILE merupakan jenis arsitektur *host* yang digunakan untuk mengembangkan sistem operasi yang akan digunakan oleh target. Gambar 19 berikut menunjukkan jendela konfigurasi *kernel linux*.



Gambar 19. Jendela konfigurasi kernel linux-2.6.11.3

Gambar 20. Pemilihan *system type* yang digunakan pada kernel 2.6.11.3



Gambar 21. Fitur *networking* kernel 2.6.11.3Gambar 22. *File system* kernel 2.6.11.3

Pada jendela tersebut dapat dilakukan pemilihan fitur-fitur yang akan digunakan pada target sistem seperti fitur *networking*, jenis *filesystem*, dan *driver device*. Lakukan penyimpanan setelah proses konfigurasi selesai

dilakukan sehingga proses akan menulis konfigurasi kernel dalam sebuah file dengan nama *config*. Kemudian keluar dari jendela konfigurasi dengan memilih menu *exit* pada jendel konfigurasi.

Langkah selanjutnya adalah melakukan pembuatan direktori *include* yang diperlukan oleh *toolchain* dan menyalin *kernel header* ke dalamnya.

```
erwin@erwin-laptop:~/ikhwan-project/kernel$ mkdir -p
${TARGET_PREFIX}/include
erwin@erwin-laptop:~/ikhwan-project/kernel$ cp -r
include/linux/ ${TARGET_PREFIX}/include

erwin@erwin-laptop:~/ikhwan-project/kernel$ cp -r
include/asm-arm/ ${TARGET_PREFIX}/include/asm

erwin@erwin-laptop:~/ikhwan-project/kernel$ cp -r
include/asm-generic/ ${TARGET_PREFIX}/include
```

direktori *asm-arm* pada perintah diatas menyesuaikan dengan jenis arsitektur target.

## **b. Instalasi *Binary Utilities***

Paket *binutils* merupakan *utilities* yang digunakan untuk memanipulasi *file object*. Langkah pertama untuk melakukan instalasi *binutils* adalah dengan mengekstrak *file binutils* yang telah di-*download* pada direktori *build-tools*. Paket *binary utilities* yang digunakan adalah *binutils-2.16.1*.

```
cd ${PRJROOT}/build-tools

erwin@erwin-laptop:~/ikhwan-project/build-tools/$ tar
xvzf binutils-2.16.1.tar.gz
```

Hasil ekstraksi adalah berupa sebuah direktori dengan nama *binutils-2.16.1*. Selanjutnya kita dapat berpindah ke direktori tersebut dan melakukan konfigurasi *binutils*.

```
erwin@erwin-laptop:~/ikhwan-project/build-tools/$ cd
build-binutils
```

```
erwin@erwin-laptop:~/ikhwan-project/build-tools/build-
binutils$ /home/erwin/ikhwan-project/build-
tools/binutils-2.16.1/configure --target=$TARGET --
prefix=${PREFIX}
```

Setelah proses konfigurasi selesai dilakukan, selanjutnya kita dapat melakukan kompilasi dan instalasi *binary utilities*.

```
erwin@erwin-laptop:~/ikhwan-project/build-tools/build-
binutils$ make
```

```
erwin@erwin-laptop:~/ikhwan-project/build-tools/build-
binutils$ make install
```

*Binary utilities* telah selesai diinstal, hal ini dapat diketahui dengan melihat isi dari direktori yang ditunjuk oleh variabel PREFIX.

```
erwin@erwin-laptop:~/ikhwan-project/build-tools/build-
binutils$ ls ${PREFIX}/bin
```

```
total 26388
```

```
arm-linux-addr2line      arm-linux-objdump
arm-linux-ar             arm-linux-ranlib
arm-linux-as             arm-linux-readelf
arm-linux-c++filt        arm-linux-size
arm-linux-ld             arm-linux-strings
arm-linux-nm             arm-linux-strip
arm-linux-objcopy
```

### c. Instalasi *Bootstrap Compiler*

Pada tahap ini penulis melakukan kompilasi *bootstrap compiler* yang hanya mendukung bahasa C dengan menggunakan paket gcc yang merupakan GNU *compiler*. Kemudian setelah *C library* selesai dilakukan kompilasi, kita akan melakukan kompilasi ulang gcc dengan dukungan bahasa C++ secara penuh. Proses instalasi dimulai dengan melakukan ekstraksi paket gcc telah di-*download* pada direktori *build-tools*. Paket gcc yang digunakan adalah gcc-2.95.3.

```
cd ${PRJROOT}/build-tools
```

```
erwin@erwin-laptop:~/ikhwan-project/build-tools/$ tar
xvzf gcc-2.95.3.tar.gz
```

Hasil ekstraksi adalah berupa sebuah direktori dengan nama gcc-2.95.3.

Selanjutnya kita dapat berpindah ke direktori tersebut dan melakukan konfigurasi gcc.

```
erwin@erwin-laptop:~/ikhwan-project/build-tools$ cd
build-boot-gcc

erwin@erwin-laptop:~/ikhwan-project/build-tools/build-
boot-gcc$ /home/erwin/ikhwan-project/build-tools/gcc-
2.95.3/configure --target=$TARGET --prefix={PREFIX} --
without-headers --with-newlib --enable-languages=c
```

Setelah proses konfigurasi selesai dilakukan, selanjutnya kita lakukan proses kompilasi gcc dengan mengetikkan perintah

```
erwin@erwin-laptop:~/ikhwan-project/build-tools/build-
boot-gcc$ make all-gcc
```

Pada proses kompilasi gcc, muncul beberapa pesan kesalahan (*error*).

Beberapa pesan kesalahan tersebut antara lain :

```
/home/erwin/ikhwan-project/build-tools/gcc-
2.95.3/gcc/config/arm/arm.c: In function
'arm_override_options':

/home/erwin/ikhwan-project/build-tools/gcc-
2.95.3/gcc/config/arm/arm.c:286: warning: assignment
discards qualifiers from pointer target type
/home/erwin/ikhwan-project/build-tools/gcc-
2.95.3/gcc/config/arm/arm.c:530: error: lvalue required
as left operand of assignment

make[1]: *** [arm.o] Error 1

make[1]: Leaving directory `/home/erwin/ikhwan-
project/build-tools/build-boot-gcc/gcc'

make: *** [all-gcc] Error 2
```

Penyebab dari *error* tersebut adalah adanya *operand* pada *file* arm.c yang tidak sesuai dengan yang dibutuhkan pada saat kompilasi. *Error* ini dapat

diselesaikan dengan mengubah beberapa baris perintah yang ada pada *file* `arm.c` yang ada pada direktori

```
/home/erwin/ikhwan-project/build-tools/gcc-
2.95.3/gcc/config/arm/
```

Baris perintah yang menandakan terjadinya *error* berada pada baris perintah 530 sehingga kita harus mengubah baris perintah tersebut yang semula adalah

```
arm_prog_mode = TARGET_APCS_32 ? PROG_MODE_PROG32 :
PROG_MODE_PROG26;
```

menjadi

```
arm_prgmode = TARGET_APCS_32 ? PROG_MODE_PROG32 :
PROG_MODE_PROG26;
```

Pesan kesalahan lain yang ditemui saat proses kompilasi adalah

```
inlined from 'collect_execute' at /home/erwin/ikhwan-
project/build-tools/gcc-2.95.3/gcc/collect2.c:1762:
/usr/include/bits/fcntl2.h:51: error: call to
'__open_missing_mode' declared with attribute error: open
with O_CREAT in second argument needs 3 arguments

make[1]: *** [collect2.o] Error 1

make[1]: Leaving directory `/home/erwin/ikhwan-
project/build-tools/build-boot-gcc/gcc'

make: *** [all-gcc] Error 2
```

Penyebab *error* ini adalah penggunaan argumen `O_CREAT` yang tidak cocok dengan fungsi `_open_missing_mode` pada *file* `collect2.c` sehingga perlu digantikan dengan argumen yang lain. *Error* kedua ini dapat diselesaikan dengan mengubah beberapa baris perintah yang ada pada *file* `collect2.c` yang ada pada direktori

```
/home/erwin/ikhwan-project/build-tools/gcc-2.95.3/gcc/
```

Baris perintah yang menandakan terjadinya *error* berada pada baris perintah 1762 sehingga kita harus mengubah baris perintah tersebut yang semula adalah

```
redir_handle = open (redir, O_WRONLY | O_TRUNC |
O_CREAT);
```

menjadi

```
redir_handle = open (redir, S_IRUSR | S_IWUSR);
```

Pesan kesalahan selanjutnya yang ditemui saat proses kompilasi adalah

Aborted

```
rm -f libgcc1.S
mv tmpgcc1.a libgcc1-asm.a
make[3]: Leaving directory `/home/erwin/ikhwan-project-
2/build-tools/build-boot-gcc/gcc'
rm -rf tmpgcc.a tmpcopy
mkdir tmpcopy
if [ xlibgcc1-asm.a != x ]; \
    then (cd tmpcopy; arm-linux-ar x ../libgcc1-asm.a); \
    else true; \
    fi
(cd tmpcopy; chmod +w * > /dev/null 2>&1)

make[2]: [stmp-multilib-sub] Error 1 (ignored)
(cd tmpcopy; arm-linux-ar x ../libgcc2.a)
(cd tmpcopy; arm-linux-ar rc ../tmpgcc.a *.o)
arm-linux-ar: *.o: No such file or directory

make[2]: *** [stmp-multilib-sub] Error 1

make[2]: Leaving directory `/home/erwin/ikhwan-project-
2/build-tools/build-boot-gcc/gcc'

make[1]: *** [stmp-multilib] Error 1

make[1]: Leaving directory `/home/erwin/ikhwan-project-
2/build-tools/build-boot-gcc/gcc'

make: *** [all-gcc] Error 2
```

Penyebab *error* diatas terjadi pada stmp-multilib-sub dan stmp-multilib serta tidak adanya *file* atau direktori arm-linux-ar yang diperlukan pada saat proses kompilasi *bootstrap compiler*. *Error* ini pun

berkaitan dengan file `libgcc2.a` dan `tmp1libgcc.a`. Penulis kesulitan menemukan solusi yang tepat untuk permasalahan *error* yang terakhir, sehingga penulis memutuskan untuk menghentikan proses perancangan dengan menggunakan metode manual. Hal ini dikarenakan jika terjadi kegagalan pada salah satu proses instalasi, maka proses selanjutnya tidak dapat dilakukan. Sebagai langkah lanjutan, penulis mencoba untuk menerapkan metode perancangan secara otomatis dengan menggunakan beberapa *tools* seperti *crosstool* dan *buildroot*.

## 2. Metode Implementasi Secara Otomatis Menggunakan *Crosstool*

*Crosstool* merupakan paket *toolchain* yang dikembangkan oleh seorang *software engineer* bernama Dan Kegel untuk memudahkan para pengembang *embedded linux system* dalam melakukan konfigurasi dan instalasi *cross-platform development toolchain*. *Crosstool* berisi *file-file* beserta dengan perintah-perintah yang digunakan untuk melakukan konfigurasi dan instalasi *toolchain*. Dalam proses instalasi *crosstool*, diperlukan koneksi internet dengan stabilitas yang cukup tinggi karena proses instalasi dilakukan dengan *men-download* paket-paket yang dibutuhkan secara langsung. Setelah proses *download* file paket yang dibutuhkan selesai, maka proses konfigurasi dan instalasi akan langsung dilakukan secara otomatis.

Proses instalasi *crosstool* diawali dengan melakukan ekstraksi *file crosstool* yang telah di-*download* dengan mengetikkan perintah berikut.

```
tar -xzf crosstool-0.43.tar.gz
```

Hasil dari proses ekstraksi adalah berupa sebuah direktori dengan nama *crosstool-0.43*. Selanjutnya kita lihat *demo script* yang ada pada direktori tersebut sesuai dengan jenis arsitektur prosesor yang akan dikembangkan. Jika arsitektur ARM, maka *file* yang harus kita lihat adalah *demo-arm.sh*. Dalam file ini dapat kita ketahui tiga variabel penting, yaitu :

```
TARBALLS_DIR=$HOME/downloads
RESULT_TOP=/opt/crosstool
GCC_LANGUAGES="c,c++,java,f77"
```

Variabel pertama mengindikasikan direktori yang digunakan sebagai tempat untuk meletakkan *file-file* paket yang akan di-*download* pada saat instalasi *crosstool*. Secara standar *file* hasil *download* akan ditempatkan pada direktori */home/downloads*, direktori ini dapat diubah sesuai dengan yang diinginkan oleh pengembang. Variabel kedua mengindikasikan direktori yang digunakan sebagai tempat instalasi *toolchain*. Direktori ini juga dapat diubah sesuai dengan keinginan pengembang sistem. Variabel terakhir mengindikasikan pilihan bahasa yang bisa dilakukan kompilasi dengan menggunakan *toolchain*.

Versi dari paket yang akan diinstal untuk arsitektur ARM dapat dilihat pada baris perintah

```
eval `cat arm.dat gcc-4.1.0-glibc-2.3.2-tls.dat` sh all.sh -
notest
```

Versi-versi dari paket tersebut merupakan versi paket yang telah diuji oleh pengembang *crosstool* sendiri. Akan tetapi kita juga dapat mengganti versi pada baris perintah tersebut sesuai dengan yang diinginkan.



Selanjutnya dengan menggunakan hak akses sebagai *root*, kita buat direktori */opt/crostool* dan membuatnya menjadi *writable* untuk *user*.

```
sudo mkdir /opt/crostool
sudo chown $USER /opt/crostool
```

Setelah beberapa pengaturan selesai dilakukan, langkah terakhir adalah menjalankan *file* *demo-arm.sh* dengan mengetikkan perintah berikut.

```
sh demo-arm.sh
```

Proses instalasi *crostool* akan berjalan secara otomatis dengan terkoneksi langsung ke internet. Beberapa paket-paket *software* yang dibutuhkan akan di-*download* secara langsung. Proses ini membutuhkan waktu yang relatif lama tergantung pada kecepatan koneksi internet dan spesifikasi *hardware host* yang digunakan.

Pada perancangan metode otomatis dengan menggunakan *crostool* ini, penulis juga menemukan berbagai macam pesan kesalahan yang serupa dengan pesan kesalahan pada saat perancangan secara manual sehingga proses instalasi pun gagal dan tidak dapat dilanjutkan.

### 3. Metode Implementasi Secara Otomatis Menggunakan *BuildRoot*

*BuildRoot* merupakan *tools* yang didesain untuk memudahkan para *engineer* dalam mengembangkan *project* yang berkaitan dengan *embedded linux system*. *Tools* ini juga sangat cocok digunakan untuk para pemula yang masih belum mengerti banyak tentang perancangan *embedded linux system*. *Tools* ini lebih mudah jika dibandingkan dengan *Crostool* karena memiliki fasilitas GUI yang memudahkan dalam melakukan konfigurasi sistem yang akan

dikembangkan berikut paket-paket *software* yang akan diinstal. Proses instalasi *buildroot* dapat dilakukan dengan dua cara yakni secara *online* maupun secara *offline*, tetapi antara keduanya sama-sama memerlukan koneksi internet dengan stabilitas yang cukup tinggi untuk melakukan *download* paket-paket *software* yang dibutuhkan. Dalam penelitian ini, penulis melakukan instalasi *buildroot* secara *online*.

Penulis menggunakan direktori *ikhwan-project* pada *home directory* sebagai direktori instalasi *buildroot*. Sebelum melakukan instalasi *buildroot*, terlebih dahulu kita melakukan ekstraksi file *buildroot* yang telah di-*download* dengan mengetikkan perintah.

```
erwin@erwin-laptop:~$ cd ikhwan-project/
erwin@erwin-laptop:~/ikhwan-project$ tar xvzf buildroot-
2009.02.tar.gz
erwin@erwin-laptop:~/ikhwan-project$ cd buildroot-2009.02/
```

*Host* yang digunakan harus dipastikan telah terinstalasi paket *library libncurses5-dev*. Jika paket tersebut belum terinstal, kita dapat melakukan instalasi dengan mengetikkan perintah

```
erwin@erwin-laptop:~$ sudo apt-get install libncurses5-dev
```

Langkah awal yang harus dilakukan dalam proses instalasi *buildroot* adalah melakukan konfigurasi untuk menentukan jenis arsitektur target, fitur-fitur sistem serta paket-paket aplikasi yang akan diaplikasikan pada *target system*. Perintah yang digunakan untuk melakukan konfigurasi adalah

```
erwin@erwin-laptop:~/ikhwan-project/buildroot-2009.02$ make
menuconfig
```

Dalam proses yang berjalan, penulis menemukan pesan kesalahan sebagai berikut.

```

Checking build system dependencies:

BUILDROOT_DL_DIR clean:           Ok
CC clean:                         Ok
CXX clean:                        Ok
CPP clean:                       Ok
CFLAGS clean:                    Ok
INCLUDES clean:                  Ok
CXXFLAGS clean:                  Ok
which installed:                  Ok
sed works:                       Ok (/bin/sed)
GNU make version '3.81':          Ok
C compiler '/usr/bin/gcc'         Ok
C compiler version '4.3.3':       Ok
C++ compiler '/usr/bin/g++'       Ok
C++ compiler version '4.3.3':     Ok
awk installed:                   Ok
bash installed:                  Ok
bison installed:                 Ok
flex installed:                  Ok
gettext installed:               FALSE

```

You must install 'gettext' on your build machine

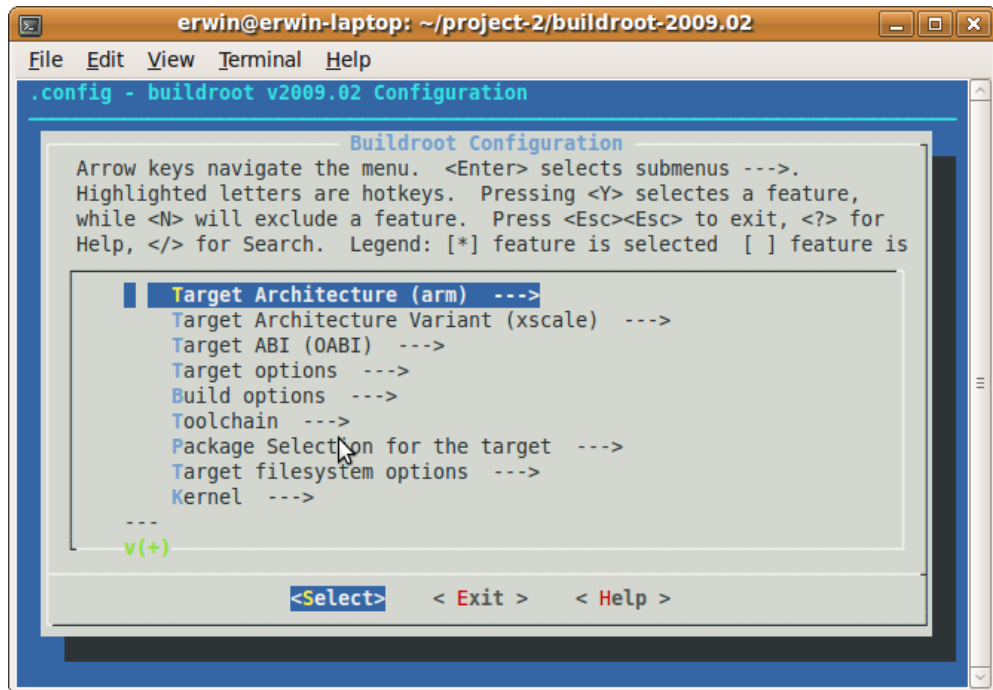
Dari pesan kesalahan yang muncul tersebut diketahui bahwa pada *host* belum terinstal aplikasi *gettext* yang ditandai dengan kondisi FALSE. Masalah tersebut dapat diselesaikan dengan melakukan instalasi aplikasi yang dimaksud.

```
erwin@erwin-laptop:~$ sudo apt-get install gettext
```

Proses konfigurasi dapat dilanjutkan dengan mengetikkan kembali perintah

```
erwin@erwin-laptop:~/ikhwan-project/buildroot-2009.02$ make
menuconfig
```

Hasil eksekusi baris perintah diatas akan memunculkan jendela konfigurasi *buildroot* yang berbasis GUI seperti ditunjukkan pada gambar 23.

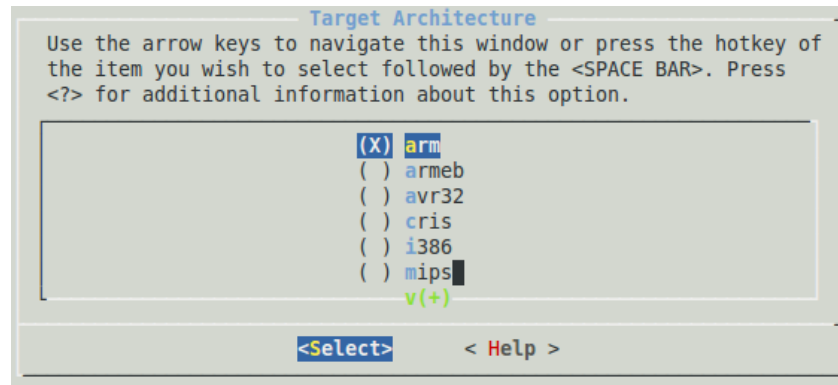


Gambar 23. Jendela utama konfigurasi *buildroot*

Beberapa hal utama yang harus diatur pada proses konfigurasi *buildroot* ini adalah :

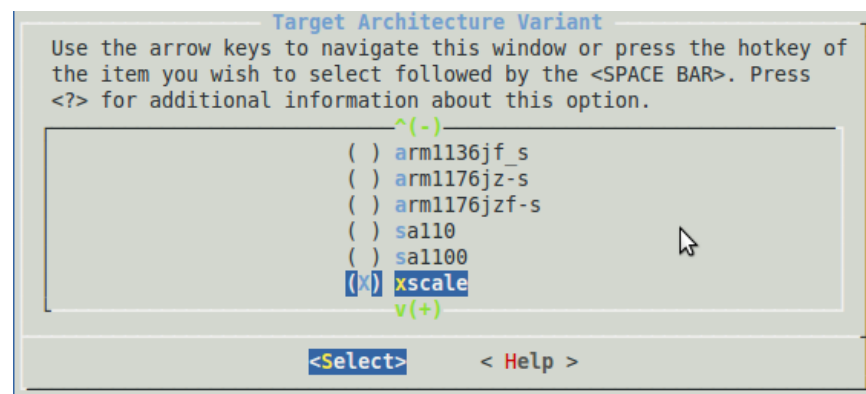
- a. Jenis dan *variant* arsitektur target,
- b. Fitur *toolchain*,
- c. Paket-paket aplikasi yang akan diinstal,

Jenis arsitektur target yang akan dikembangkan dapat dipilih melalui menu *Target Architecture* sehingga ditampilkan beberapa pilihan jenis arsitektur seperti ditunjukkan gambar 24 berikut ini.



Gambar 24. Target architecture

Sedangkan pemilihan *variant* dari jenis arsitektur yang telah ditentukan melalui menu *Target Architecture Variant* sehingga ditampilkan beberapa pilihan *variant* untuk jenis arsitektur tertentu seperti ditunjukkan gambar 25.



Gambar 25. Target architecture variant

Langkah konfigurasi selanjutnya adalah melakukan pemilihan fitur *toolchain* yang digunakan untuk perancangan *target system*. Minimal fitur yang harus diaktifkan adalah fitur untuk *compiler c++*.

```

Toolchain
er> selects submenus --->. Highlighted letters are hotkeys. Pressing <Y>
> to exit, <?> for Help, </> for Search. Legend: [*] feature is selected

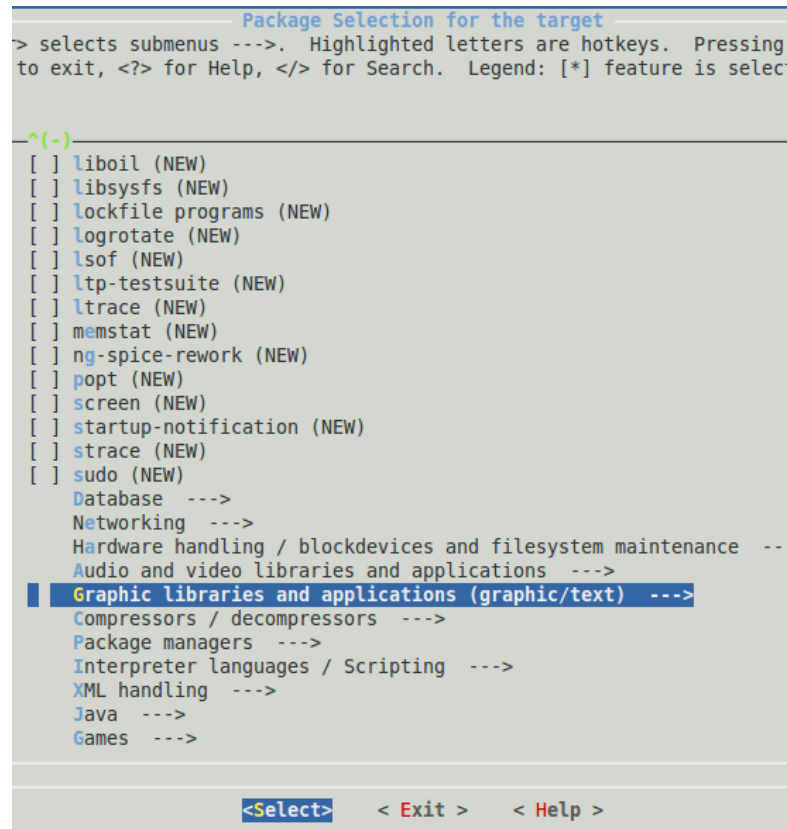
^(-)
[*] Build/install a shared libgcc? (NEW)
*** Ccache Options ***
[ ] Enable ccache support? (NEW)
*** Gdb Options ***
[ ] Build gdb debugger for the Target (NEW)
[ ] Build gdb server for the Target (NEW)
[ ] Build gdb for the Host (NEW)
*** Common Toolchain Options ***
[ ] Enable large file (files > 2 GB) support? (NEW)
[ ] Enable IPv6 (NEW)
[ ] Enable RPC (NEW)
[*] Enable locale/gettext/il8n support?
[ ] Use pregenerated locale data? (NEW)
[*] Enable WCHAR support
[ ] Use software floating point by default (NEW)
Thread library implementation (linuxthreads (stable/old)) --->
[*] Build/install c++ compiler and libstdc++?
(-Os -pipe) Target Optimizations (NEW)
[ ] Enable elf2flt support? (NEW)
[ ] Run mklibs on the built root filesystem (NEW)
[ ] Install sstrip for the target system (NEW)
[ ] Install sstrip for the host/build system (NEW)
[ ] Enable multilib support? (NEW)
[ ] Use ARM Vector Floating Point unit (NEW)
[*] Include target utils in cross toolchain (NEW)

<Select> < Exit > < Help >

```

Gambar 26. Fitur-fitur *Toolchain*

Langkah selanjutnya adalah pemilihan paket-paket aplikasi yang akan diimplementasikan pada target. Paket aplikasi yang disertakan dalam *buildroot* harus diseleksi terlebih dahulu menyesuaikan dengan sistem yang akan dikembangkan karena jumlahnya yang cukup banyak. Akan tetapi kita juga dapat mengimplementasikan semua paket aplikasi yang disediakan, dengan konsekuensi pada penambahan kapasitas *file* yang dihasilkan dari proses ini serta proses instalasi yang akan bertambah lama. Pemilihan paket-paket aplikasi dilakukan melalui menu *Package Selection for The Target* seperti pada gambar 27 berikut.



Gambar 27. Package selection for the target

Tahapan konfigurasi selanjutnya adalah konfigurasi *filesystem* untuk target dan pemilihan jenis *bootloader* yang akan diterapkan pada target. Pada *embedded pc CSB625*, *bootloader* yang digunakan adalah Micromonitor sedangkan secara standar untuk arsitektur ARM, *bootloader* yang digunakan adalah U-boot. Dengan adanya perbedaan ini, maka harus dilakukan proses *patching* pada *buildroot* agar mendukung *bootloader* Micromonitor. Proses patch dilakukan dengan menggunakan kode sebagai berikut.

```
diff -purN buildroot/target/Config.in buildroot-
umon/target/Config.in
--- buildroot/target/Config.in 2008-03-31 00:15:26.000000000
-0700
+++ buildroot-umon/target/Config.in 2008-03-31
22:22:09.000000000 -0700
@@ -19,6 +19,7 @@ source "target/x86/grub/Config.in"
    source "target/x86/syslinux/Config.in"
```

```

source "target/powerpc/yaboot/Config.in"
source "target/u-boot/Config.in"
+source "target/umon/Config.in"
endmenu

menu "Kernel"
diff      -purN      buildroot/target/Makefile.in      buildroot-
umon/target/Makefile.in
--- buildroot/target/Makefile.in 2008-03-31  00:15:26.000000000
-0700
+++ buildroot-umon/target/Makefile.in 2008-03-31
22:22:30.000000000 -0700
@@ -15,6 +15,10 @@ ifeq ($(strip $(BR2_TARGET_UBOOT)),y)
    include target/u-boot/Makefile.in
endif

+ifeq ($(strip $(BR2_TARGET_UMON)),y)
+include target/umon/Makefile.in
+endif
+
# and finally build the filesystems/tarballs
include target/*/*.mk

diff      -purN      buildroot/target/umon/Config.in      buildroot-
umon/target/umon/Config.in
--- buildroot/target/umon/Config.in 1969-12-31
16:00:00.000000000 -0800
+++ buildroot-umon/target/umon/Config.in 2008-03-31
22:18:49.000000000 -0700
@@ -0,0 +1,20 @@
+config BR2_TARGET_UMON
+ bool "Micromonitor Boot Loader"
+ default n
+ help
+   Build uMon bootloader.
+
+config BR2_TARGET_UMON_PORT
+ string "port name"
+ depends on BR2_TARGET_UMON
+ default "$(BOARD_NAME)"
+ help
+   uMon port name. This is the name of the directory under
umon_ports.
+
+config BR2_TARGET_UMON_CUSTOM_PATCH
+ string "custom patch"
+ depends on BR2_TARGET_UMON
+ help
+   If your board requires a custom patch, add the path to the
file here.
+   Most users may leave this empty.
+
diff      -purN      buildroot/target/umon/Makefile.in      buildroot-
umon/target/umon/Makefile.in
--- buildroot/target/umon/Makefile.in 1969-12-31
16:00:00.000000000 -0800
+++ buildroot-umon/target/umon/Makefile.in 2008-03-31
22:18:49.000000000 -0700
@@ -0,0 +1,66 @@

```



```

#####
+#
+# umon
+#
#####
+UMON_VERSION:=sep8_2007
+UMON_SOURCE:=umon_${UMON_VERSION}.tgz
+UMON_SITE:=http://microcross.com
+UMON_PORT:=$(strip $(subst ",,,$(BR2_TARGET_UMON_PORT)))
+UMON_DIR:=$(PROJECT_BUILD_DIR)/umon/umon_ports/${UMON_PORT}
+UMON_HOST_DIR:=$(PROJECT_BUILD_DIR)/umon/umon_main/host
+UMON_PATCH_DIR:=$(PROJECT_BUILD_DIR)/umon-patches
+UMON_CAT:=$(ZCAT)
+UMON_BIN:=boot.bin
+UMON_TOP:=$(PROJECT_BUILD_DIR)/umon/umon_main
+# this is a nasty hack to get the PLATFORM variable from the
makefile
+UMON_PLATFORM:=$( $(grep '^PLATFORM.*=' $(UMON_DIR)/makefile |
sed 's/^PLATFORM.*=@@')
+
+$(DL_DIR)/$(UMON_SOURCE):
+ $(WGET) -P $(DL_DIR) $(UMON_SITE)/$(UMON_SOURCE)
+
+$(UMON_DIR)/.unpacked: $(DL_DIR)/$(UMON_SOURCE)
+ $(UMON_CAT) $(DL_DIR)/$(UMON_SOURCE) \
+ | tar -C $(PROJECT_BUILD_DIR) $(TAR_OPTIONS) -
+ touch $@
+
+$(UMON_DIR)/.patched: $(UMON_DIR)/.unpacked
+ifneq ($(strip $(BR2_TARGET_UMON_CUSTOM_PATCH)), "")
+ @mkdir -p $(UMON_PATCH_DIR)
+ cp -dpr $(BR2_TARGET_UMON_CUSTOM_PATCH) $(UMON_PATCH_DIR)
+ toolchain/patch-kernel.sh $(PROJECT_BUILD_DIR)/umon
$(UMON_PATCH_DIR) *.patch
+endif
+ touch $@
+
+$(UMON_DIR)/build_${UMON_PLATFORM}/${UMON_BIN}:
$(UMON_DIR)/.patched
+ $(MAKE) -C $(UMON_HOST_DIR) UMON_TOP=$(UMON_TOP)
OSTYPE=linux install
+ $(MAKE) $(TARGET_CONFIGURE_OPTS) -C $(UMON_DIR)
UMONTOP=$(UMON_TOP)
+
+$(BINARIES_DIR)/$(UMON_BIN):
$(UMON_DIR)/build_${UMON_PLATFORM}/${UMON_BIN}
+ cp $(UMON_DIR)/build_${UMON_PLATFORM}/${UMON_BIN}
$(BINARIES_DIR)/$(UMON_BIN)
+
+umon: $(BINARIES_DIR)/$(UMON_BIN)
+
+umon-clean:
+ $(MAKE) -C $(UMON_DIR) clean
+
+umon-dirclean:
+ rm -rf $(UMON_DIR)
+
+umon-source: $(DL_DIR)/$(UMON_SOURCE)
+

```

```

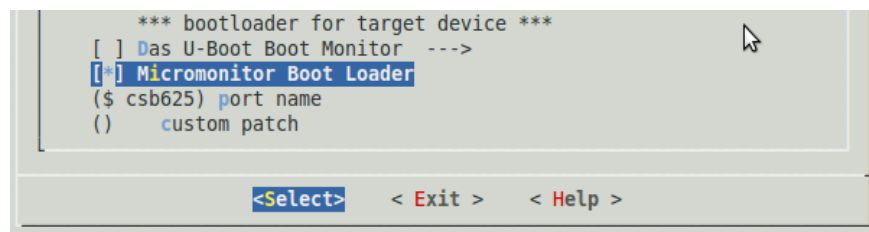
#####
+#
+# Toplevel Makefile options
+#
#####
+ifeq ($(strip $(BR2_TARGET_UMON)),y)
+TARGETS+=umon
+endif
+
+umon-status:
+ @echo
+ @echo BR2_TARGET_UMON_PORT = $(BR2_TARGET_UMON_PORT)
+ @echo BR2_TARGET_UMON_CUSTOM_PATCH = $(BR2_TARGET_UMON_CUSTOM_PATCH)
+ @echo
+ @exit 0

```

Kode-kode tersebut dibuat dalam sebuah *file* dengan ekstensi *.patch*, sebagai contoh *umon\_buildroot.patch*. Pada terminal *console*, *pointer* diarahkan ke direktori *buildroot* dan lakukan proses *patching* dengan mengetikkan perintah sebagai berikut.

```
patch -p1 < /home/erwin/Desktop/Umon/umon_buildroot.patch
```

Dengan menggunakan *patch* tersebut maka pada konfigurasi *buildroot* akan muncul pilihan *bootloader* Micromonitor seperti ditunjukkan pada gambar 28 berikut.

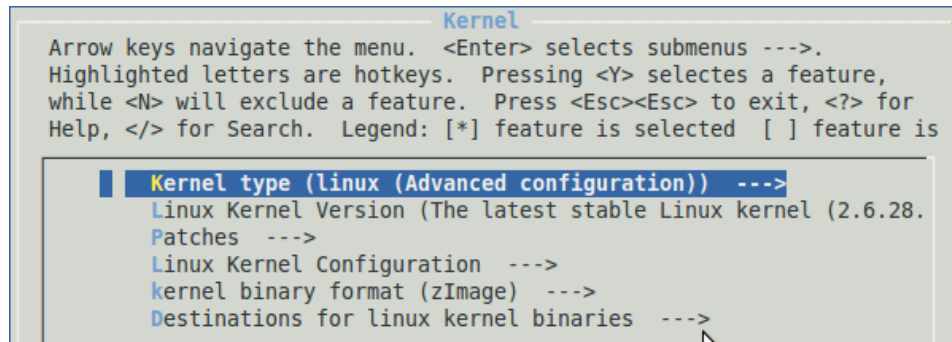


Gambar 28. Pilihan jenis *bootloader*

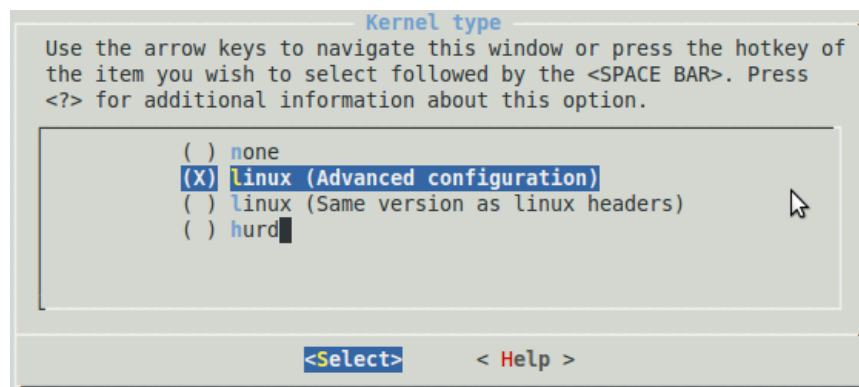
*Port name* disesuaikan dengan *platform* yang digunakan.

Tahapan terakhir dalam konfigurasi *buildroot* adalah pemilihan versi kernel linux dengan mengatur *kernel type* pada posisi *Linux Advanced Configuration*

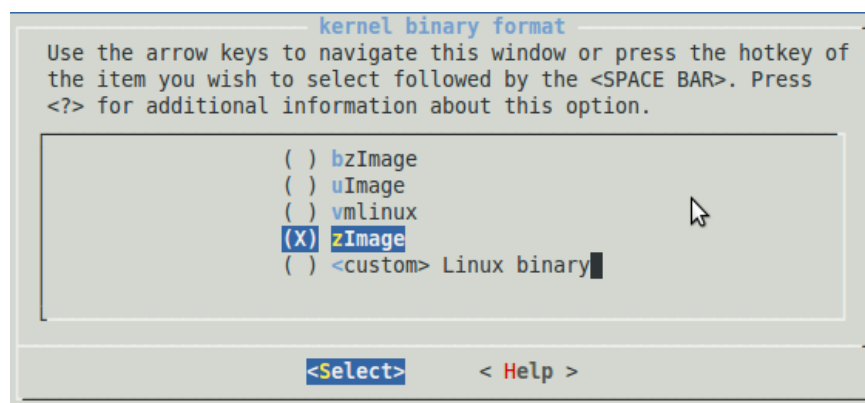
dan menentukan versi yang akan digunakan serta format *kernel binary* yang akan digunakan oleh target. Secara standar, arsitektur ARM menggunakan format *kernel binary* zImage.



Gambar 29. Konfigurasi kernel

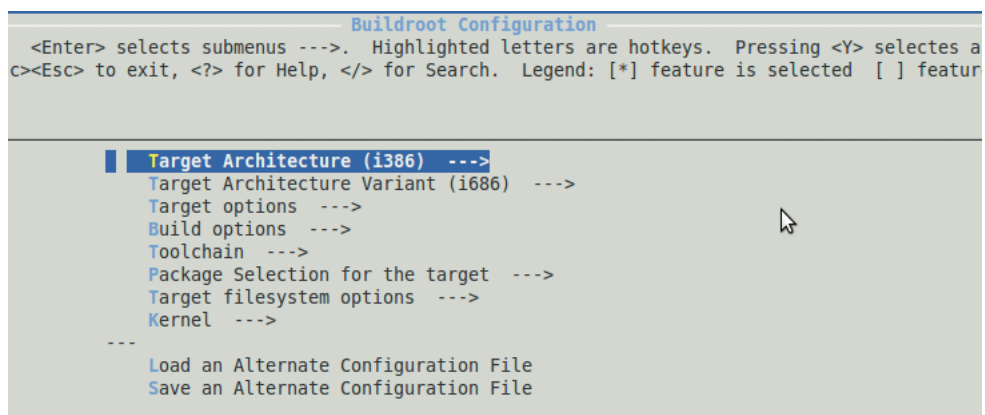


Gambar 30. Kernel type



Gambar 31. Format kernel binary

Setelah proses konfigurasi selesai dilakukan, maka lakukan proses penyimpanan konfigurasi dengan memilih menu *Save an alternate Configuration File* pada jendela utama konfigurasi *buildroot*. Kemudian pilih menu *exit* untuk mengakhiri proses konfigurasi dan menutup jendela konfigurasi *buildroot*.



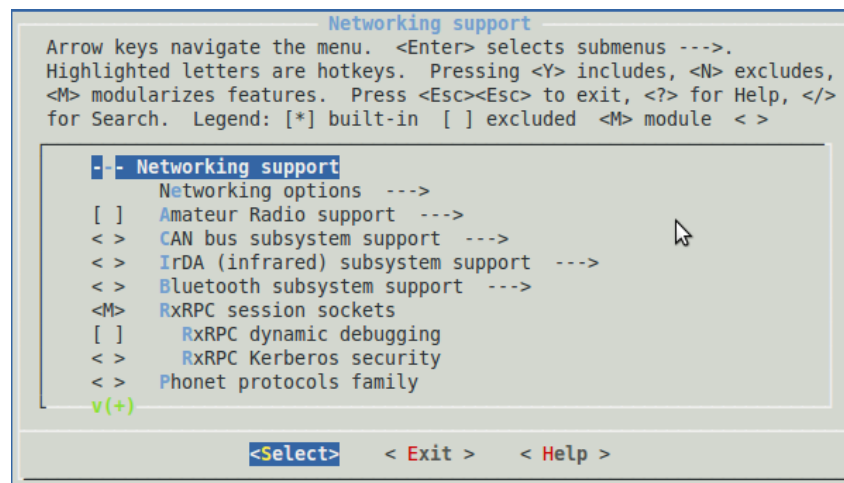
Gambar 32. *Save an alternate configuration file*

Dengan konfigurasi yang telah ditentukan, selanjutnya kita siap untuk melakukan kompilasi dan instalasi fitur-fitur serta paket-paket aplikasi yang telah dipilih dengan mengetikkan perintah berikut pada *console*.

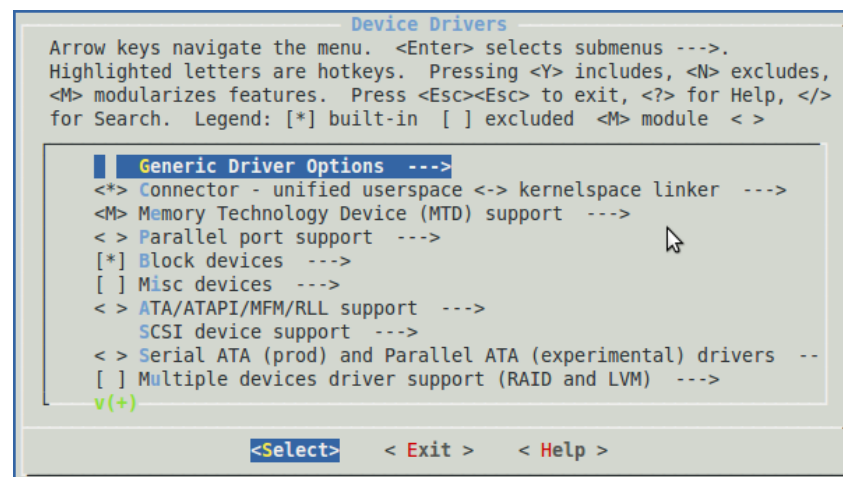
```
erwin@erwin-laptop:~/ikhwan-project/buildroot-2009.02$ make
```

Proses kompilasi dan instalasi akan berjalan secara otomatis termasuk kompilasi dan instalasi *kernel linux* yang akan digunakan untuk target. Proses ini membutuhkan waktu yang cukup lama, karena paket-paket aplikasi yang akan diinstal harus di-*download* terlebih dahulu kemudian dilakukan ekstraksi, konfigurasi dan instalasi.

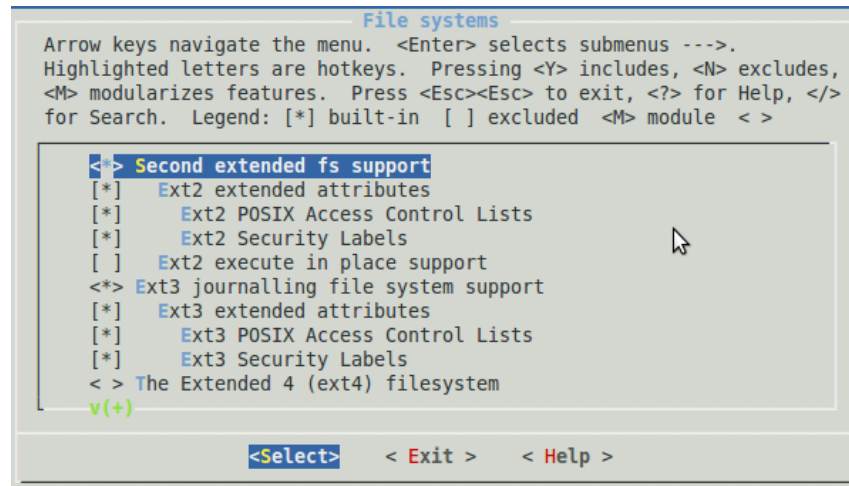
Ketika proses kompilasi dan instalasi sampai pada konfigurasi kernel maka kita akan diminta untuk mengkonfigurasi kernel yang akan digunakan untuk target. Fitur-fitur apa saja yang akan diaktifkan, jenis *device driver* yang akan digunakan, format *filesystem* yang didukung serta pengaturan lain jika diperlukan.



Gambar 33. Konfigurasi fitur *networking* kernel 2.6.28.2



Gambar 34. Konfigurasi *device driver* kernel 2.6.28.2



Gambar 35. Konfigurasi *file system* kernel 2.6.28.2

Setelah proses konfigurasi kernel selesai dilakukan dan telah dilakukan penyimpanan, maka proses kompilasi dan instalasi akan dilanjutkan kembali.

*BuildRoot* menerapkan sistem *stamping*, sebagai penanda terhadap sebuah fitur atau aplikasi yang telah selesai diinstal. Dengan adanya sistem *stamp* ini, kita dapat melakukan penundaan proses tanpa harus mengulang kembali dari awal. Ketika proses berjalan kembali, maka bagian yang sudah selesai dilakukan konfigurasi dan instalasi akan dilewati dan dilanjutkan pada bagian yang masih tersisa.

Pada tahap terakhir proses kompilasi dan instalasi sistem *embedded* menggunakan *buildroot* ini, penulis menemukan pesan kesalahan yang belum ditemukan solusinya sehingga proses terhenti dan tidak dapat dilanjutkan.

Pesan kesalahan yang ditemui adalah sebagai berikut

```
/home/erwin/project-2/buildroot-
2009.02/build_arm/staging_dir/usr/bin/./lib/gcc/arm-linux-
uclibc/4.3.2/libgcc.a(_dvmld_lnx.o): In function `__div0':
```

```
/home/erwin/project-2/buildroot-
2009.02/toolchain_build_arm/gcc-
```

```
4.3.2/libgcc/../../gcc/config/arm/lib1funcs.asm:1079:   undefined
reference to `raise'
```

```
make[1]: *** [/boot.elf] Error 1
make[1]: Leaving directory `/home/erwin/project-2/buildroot-
2009.02/project_build_arm/uclibc/umon/umon_ports/csb625'
```

```
make:          ***          [/home/erwin/project-2/buildroot-
2009.02/project_build_arm/uclibc/umon/umon_ports/csb625/build_
$(grep] Error 2
```

Penyebab dari *error* tersebut diatas berada pada fungsi `__div0` yang berada pada file `libgcc.a` yang berhubungan dengan fungsi `raise` pada file `lib1funcs.asm`. Fungsi `__div0` merupakan sebuah fungsi yang mendefinisikan pembagian dengan angka 0 yang harus dihindari dalam perhitungan maupun program. Penulis sudah mencoba melakukan proses *patching* dengan menggunakan kode-kode program untuk menyelesaikan masalah diatas. Akan tetapi usaha tersebut tidak memberikan solusi apapun terhadap *error* tersebut. Kode program untuk proses *patching* yang digunakan adalah sebagai berikut.

```
--- uClibc-0.9.28/ldso/include/dl-string.h.vanilla 2006-04-02
17:18:31.363080250 +0200
+++ uClibc-0.9.28/ldso/include/dl-string.h 2006-04-02
17:18:58.144754000 +0200
@@ -228,7 +228,8 @@
     char temp;
     do_rem(temp, i, 10);
     *--p = '0' + temp;
-    i /= 10;
+    i /= 2;
+    i *= -(-1ul / 5ul);
 }
```

Secara ideal jika proses instalasi berjalan dengan sempurna tanpa adanya pesan kesalahan maka informasi terakhir yang dapat dilihat sebagai indikator bahwa proses telah selesai adalah

```
/buildroot-2009.02/binaries/uclibc/rootfs.arm.ext2
/project_build_arm/uclibc/.fakeroot*
```

Informasi tersebut menandakan bahwa proses pembangunan sistem *embedded linux* untuk target telah selesai dilakukan. *File-file* yang dihasilkan dari proses tersebut adalah berupa *root filesystem* dalam format *filesystem ext2* (`rootfs.arm.ext2`) dan *kernel image* (`zImage`) yang siap digunakan oleh target.