

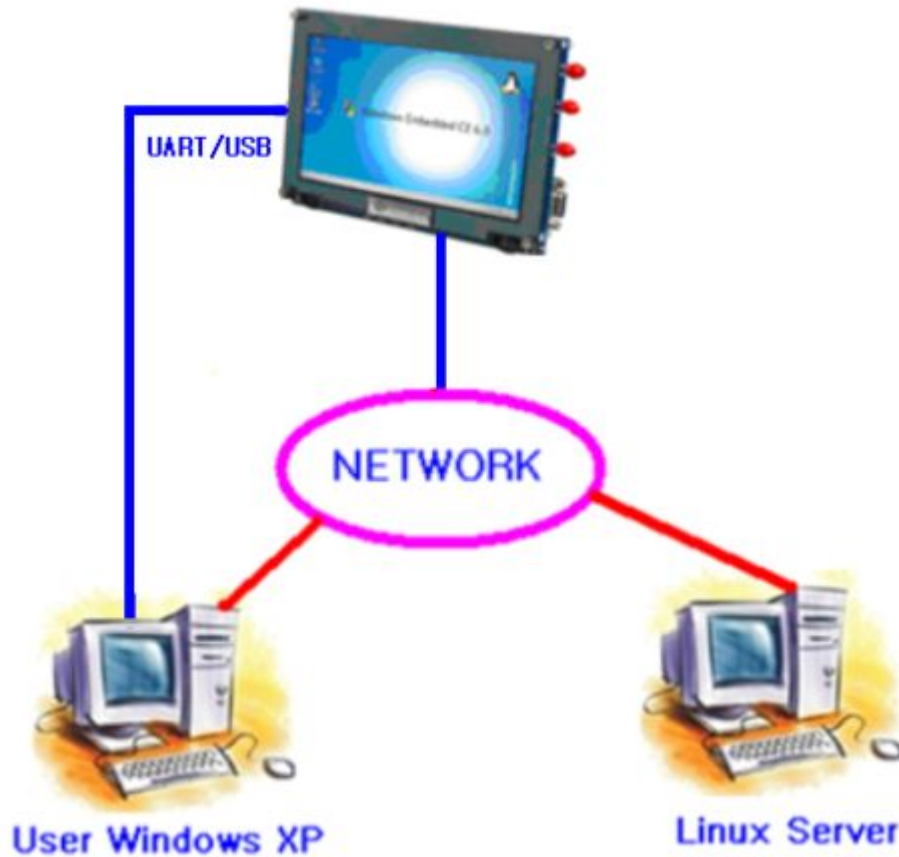
MV6410-LCD Linux 활용 가이드



(주) 마이크로비전

1. 리눅스 환경 설정

개발을 하기 앞서 먼저 환경 설정을 해야 한다. 그 중 가장 중요한 네트워크 설정부분 과 GCC 설정 방법을 먼저 설명하겠다. 참고로 당사는 페도라6 리눅스로 작업했다.



위의 그림처럼 리눅스 서버와 작업 할 유저 PC 가 네트워크 상태로 연결되어 있어야 하고, 역시 타겟 보드도 모니터링 할 수 있게 시리얼(UART)이 연결 되어있어야 한다.

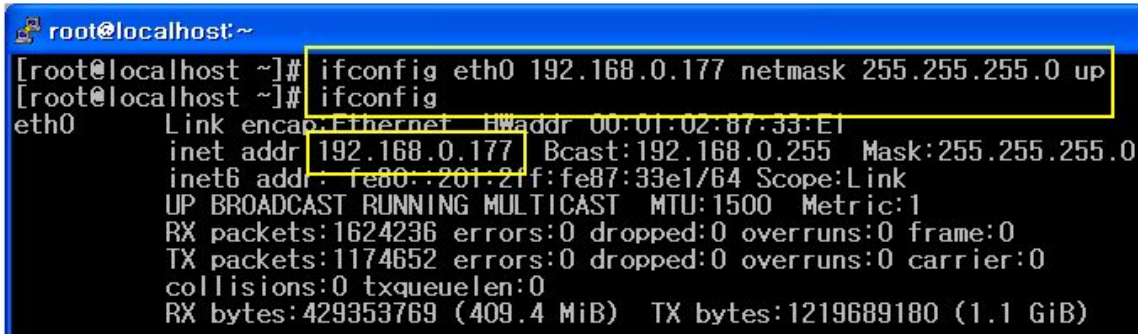
환경 설정 구축 목록은 다음과 같다.

- ▶ Linux Server IP 설정
- ▶ minicom 설정
- ▶ tftp 서버 설정
- ▶ nfs 서버 설정
- ▶ GCC 환경 구축

1) Linux Server IP 설정

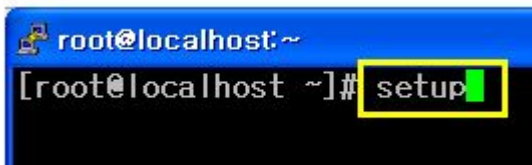
다음과 같이 명령어를 입력하면 IP 설정 할 수 있다. 또한 “ifconfig” 명령어를 통해 IP 주소를 확인 할 수 있다.

ifconfig eth0 192.168.0.177 netmask 255.255.255.0 up

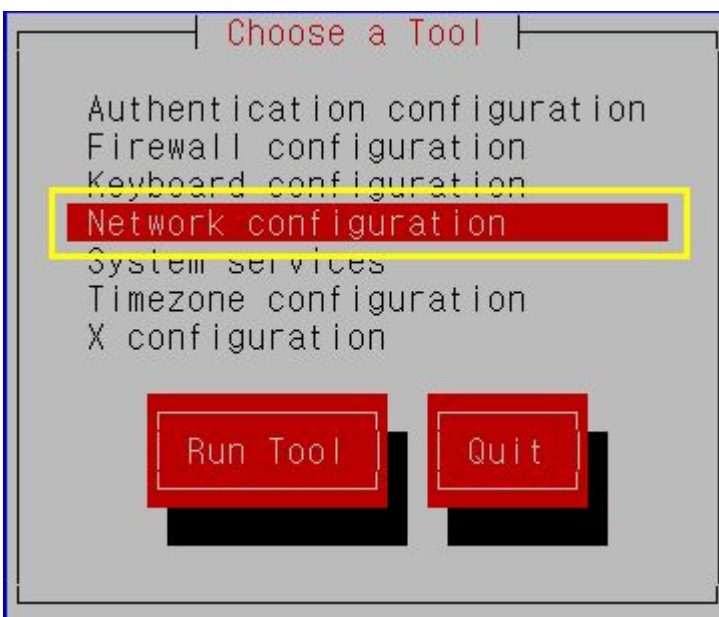


또한 “setup” 명령어를 이용해서 설정 할 수 도 있다.

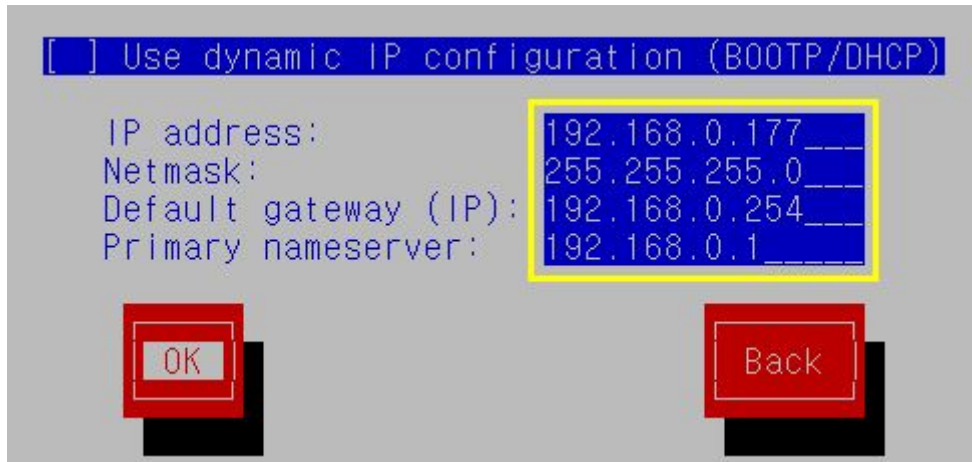
setup



“Network configuration” 선택



IP 작성 후 “OK” 선택

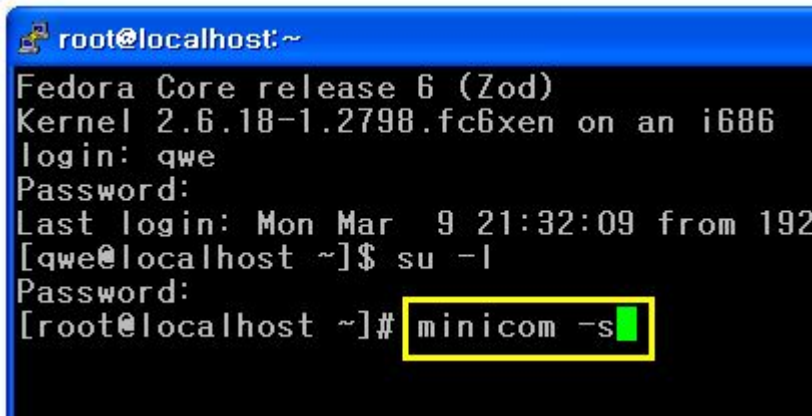


2) minicom 설정

리눅스 상에서 RS-232C 통해 타겟보드와 PC 간의 모니터링을 할 수 있게 해주는 프로그램이다. 윈도우에서 하이퍼 터미널 과 같은 프로그램이라고 생각하면 된다.

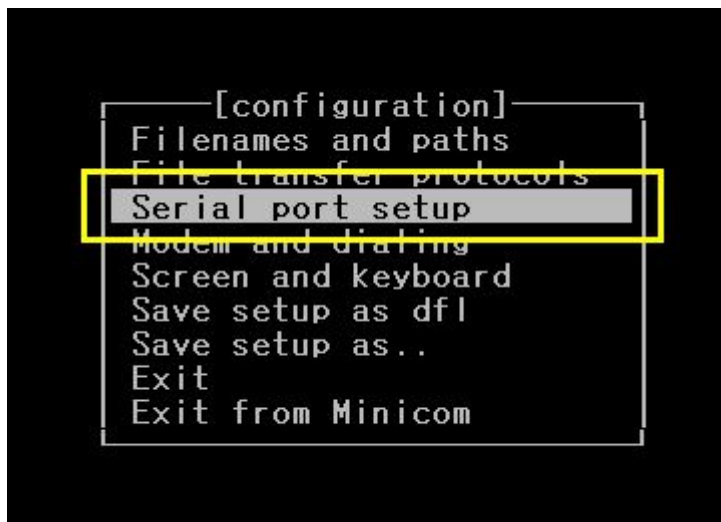
셸프롬프트 상태에서 “minicom -s” 실행 한다.

```
# minicom -s
```



```
root@localhost:~  
Fedora Core release 6 (Zod)  
Kernel 2.6.18-1.2798.fc6xen on an i686  
login: qwe  
Password:  
Last login: Mon Mar  9 21:32:09 from 192  
[qwe@localhost ~]$ su -  
Password:  
[root@localhost ~]# minicom -s
```

“Serial port setup” 선택한다.



```
[configuration]  
Filenames and paths  
File transfer protocols  
Serial port setup  
Modem and dialing  
Screen and keyboard  
Save setup as df1  
Save setup as..  
Exit  
Exit from Minicom
```

밑에 있는 옵션대로 설정을 한다.

설정 방법은 다음과 같다. 만약 Serial Device 를 설정 하고 싶으면 “A” 입력하고, 설정한 뒤 “Enter” 키를 누르면 된다.

```
A - Serial Device      : /dev/ttyS0
B - Lockfile Location  : /var/lock
C - Callin Program     :
D - Callout Program    :
E - Bps/Par/Bits       : 115200 8N1
F - Hardware Flow Control : No
G - Software Flow Control : No

Change which setting? █

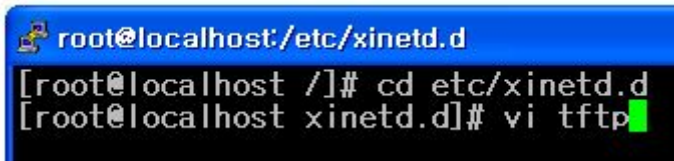
Screen and keyboard
Save setup as dfl
Save setup as..
Exit
Exit from Minicom
```

3) tftp 서버

tftp를 설정하면 u-boot, zImage, File System 이미지를 이더넷을 이용해서 타겟 보드에 올릴 수 있다.

```
# cd etc/xinetd.d
```

```
# vi tftp
```



수정 전

```
service tftp
{
    disable = yes
    socket_type      = dgram
    protocol        = udp
    wait            = yes
    user             = root
    server           = /usr/sbin/in.tftpd
    server_args     = -s /tftpboot
    per_source      = 11
    cps              = 100 2
    flags           = IPv4
}
```

수정 후

```
service tftp
{
    disable = no
    socket_type      = dgram
    protocol        = udp
    wait            = yes
    user             = root
    server           = /usr/sbin/in.tftpd
    server_args     = -s /tftpboot
    per_source      = 11
    cps              = 100 2
    flags           = IPv4
}
```

위의 그림 처럼 “disable = no” 바꾸어 주어야 tftp가 구동된다.

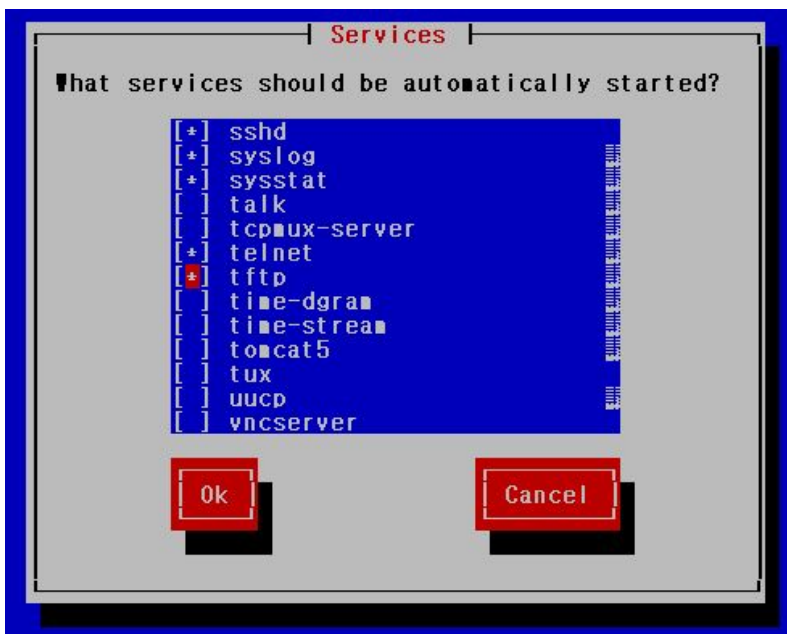
다음은 “setup” 명령을 통해 tftp 설정 확인 해보겠다.

```
# setup
```

“System services” 선택한다.



“tftp” 선택한다.



“OK”클릭 후 다음과 같이 xinetd 데몬을 수행한다.

```
# service xinetd restart
```


4) nfs 설정

nfs는 타겟 보드와 리눅스 서버 간의 폴더를 공유해 자유롭게 파일을 수정 및 복사 할 수 있는 프로그램이다.

먼저 리눅스 서버 쪽 설정이 해주어야 한다.

다음과 같은 순서 대로 명령어를 입력한다.

- # mkdir nfs
- # chmod 777 nfs
- # chown nobody nfs
- # cd etc
- # vi exports

```

root@localhost:/etc
[ root@localhost / ]# mkdir nfs
[ root@localhost / ]# chmod 777 nfs
[ root@localhost / ]# chown nobody nfs
[ root@localhost / ]# cd etc
[ root@localhost etc ]# vi exports
  
```

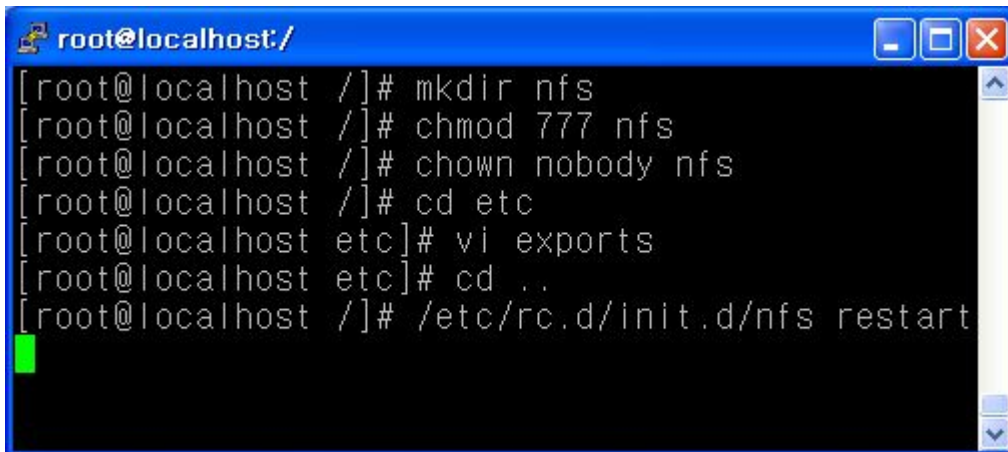
“/nfs 192.168.0.*(rw,sync,no_root_squash)” 기입한다. /nfs 이후는 ‘탭’을 이용해 이동한 후 192.168.0.*(rw,sync,no_root_squash) 기입해준다.

```

root@localhost:/etc
/nfs 192.168.0.*(rw,sync,no_root_squash)
~
~
~
~
~
~
"exports" 1L, 41C          1,1          All
  
```

다음 명령어를 이용해 “nfs” 데몬을 다시 구동 시킨다.

```
# /etc/rc.d/init.d/nfs restart
```



```
root@localhost:/  
[root@localhost ~]# mkdir nfs  
[root@localhost ~]# chmod 777 nfs  
[root@localhost ~]# chown nobody nfs  
[root@localhost ~]# cd etc  
[root@localhost etc]# vi exports  
[root@localhost etc]# cd ..  
[root@localhost ~]# /etc/rc.d/init.d/nfs restart  
█
```

타겟보드 설정은 다음과 같이 해준다.

```
[root@glibc root]# ls
bplay*          fbcam*          rt73.ko         s3c-tvscaler.ko
brec*          mpg123*        s3c-tvenc.ko   tv_test*
[root@glibc root]# cd ..
[root@glibc /]# ls
Qtopia/  dev/      home/  linuxrc@  opt/     root/     sys/     usr/
bin/     etc/     lib/   mnt/     proc/    sbin/    tmp/    var/
[root@glibc /]# cd Qtopia/
[root@glibc ~]#
```

위에 그림 처럼 보드를 처음 부팅하면 /root 폴더가 나오고 “cd ..” 명령어를 이용하여 상위 디렉토리로 가면 여러 가지 폴더가 나오는데, 현재 여기 부분은 Read Only 이므로 저널링으로 작업한 “Qtopia” 폴더에서 nks 마운트를 해야 읽고, 쓰기가 가능하다. 먼저 “Qtopia” 폴더에서 “mkdir” 명령어를 이용해 nfs 폴더를 생성하고 “chmod” 명령어를 이용해 모든 권한을 준다.

마운트 명령은 다음과 같다.

mount -t nfs -o nolock 리눅스서버IP:/nfs /타겟보드 nfs 폴더

mount -t nfs -o nolock 192.168.0.177:/nfs /Qtopia/nfs

```
[root@glibc ~]# pwd
~/Qtopia
[root@glibc ~]# mkdir nfs
[root@glibc ~]# chmod 777 nfs/
[root@glibc ~]# mount -t nfs -o nolock 192.168.0.177:/nfs /Qtopia/nfs
[root@glibc ~]# cd nfs
[root@glibc nfs]# ls
bash_profile*
[root@glibc nfs]#
```

<MV6410-LCD>

```
root@localhost:/nfs
[root@localhost nfs]# ls
bash_profile
[root@localhost nfs]#
```

<리눅스 PC NFS>

위 그림처럼 리눅스 서버 “nfs” 폴더가 타겟 보드 Qtopia/nfs에 마운트 되어 “bash_profile” 파일이 공유된 모습을 볼 수 있다.

5) GCC 환경 구축

일반 PC 데스크 탑 x86 Linux 에서 컴파일 하게 되면 그 컴퓨터에 맞는 바이너리코드가 생성된다. 이 바이너리코드는 타겟보드에 저장할 수 있는 공간이나 메모리 부분이 틀리기 때문에 타겟용으로 개발하기 위해서는 x86 컴퓨터에서 ARM 이 사용할 수 있는 바이너리 코드를 만들기 위해 크로스 컴파일러가 필요하다.

먼저 /usr/local/arm 폴더를 만들고 CD 안에 Sources\Linux\toolchain 있는 “4.3.1-eabi-armv6-mv20081010.tar.gz” 파일을 리눅스 PC /usr/local/arm 복사한다.

명령어는 다음과 같다.

```
# mkdir -p /usr/local/arm
# tar xvf 4.3.1-eabi-armv6-mv20081010.tar
# mv 4.3.1 /usr/local/arm/
# export PATH=$PATH:/usr/local/arm/ 4.3.1-eabi-armv6/usr/bin/arm-linux-
```

그리고 반드시 **bash_profile** 안에 PATH 경로를 확인 해 준다.

명령어는 다음과 같다

```
# vi ~/.bash_profile
PATH=$PATH:$HOME/bin:/usr/local/arm/4.3.1-eabi-armv6/usr/bin/
LD_LIBRARY_PATH=/usr/local/arm/4.3.1-eabi-armv6/gmp/lib:/usr/local/arm/4.3.1-eabi-armv6/mpfr/lib
export PATH LD_LIBRARY_PATH
unset USERNAME
LANG=en
```

말에 설정은 당사 리눅스 PC 의 환경 설정이다.

```
PATH=$PATH:$HOME/bin:/usr/local/arm/4.3.1-eabi-armv6/usr/bin
LD_LIBRARY_PATH=/usr/local/arm/4.3.1-eabi-armv6/gmp/lib:/usr/local/arm/4.3.1-eabi-
armv6/mpfr/lib
export PATH LD_LIBRARY_PATH
unset USERNAME
LANG=en
```

4.3.x . GCC는 반드시 gmp 와 mpfr 경로를 설정해 주어야 한다.

For example :

```
LD_LIBRARY_PATH=/usr/local/arm/4.3.1-eabi-armv6/gmp/lib:/usr/local/arm/4.3.1-eabi-  
armv6/mpfr/lib  
export PATH LD_LIBRARY_PATH
```

또한 절대 “LD_LIBRARY_PATH” 이 이름을 바꾸어서는 안된다. 설정이 끝나면 환경 적용을 해준다.

명령어는 다음과 같다.

```
# source ~/.bash_profile
```

2. 자주 사용되는 리눅스 명령어

파일보기

ls, # ls -a # ll

```

root@localhost:~/mv6410
[root@localhost mv6410]# ls
bash_profile
[root@localhost mv6410]# ls -a
bash_profile
[root@localhost mv6410]# ll
total 4
-rwxrwxr-- 1 root root 528 Mar  5 00:11 bash_profile
[root@localhost mv6410]#

```

현재 접속된 사용자

who

```

root@localhost:/
[root@localhost /]# who
root pts/1 Feb 26 19:09 (:0.0)
qwe pts/2 Mar 11 19:29 (192.168.0.112)
qwe pts/3 Mar 11 20:46 (192.168.0.112)
[root@localhost /]#

```

현재 디렉토리 위치

pwd

```

root@localhost:/
[root@localhost /]# pwd
/
[root@localhost /]#

```

파일 위치 보기

which xxx

```

root@localhost:/
[root@localhost /]# which clear
/usr/bin/clear
[root@localhost /]#

```

폴더 생성 및 삭제

mkdir xxx <- 폴더 생성

rmdir xxx <- 폴더 삭제

```

root@localhost:~
[root@localhost ~]# mkdir mv6410
[root@localhost ~]# ll
total 100
drwxr-xr-x  3 root root  4096 Feb 26 19:09 Desktop
-rw-----  1 root root  1076 Oct 29 01:16 anaconda-ks
-rw-r--r--  1 root root 55933 Oct 29 01:15 install.log
-rw-r--r--  1 root root  7826 Oct 29 01:14 install.log
-rw-r--r--  1 root root    35 Mar 10 00:43 minicom.log
drwxr-xr-x  2 root root  4096 Mar 11 22:35 mv6410
drwxr-xr-x 11 root root  4096 Mar 11 20:08 speedwee
[root@localhost ~]# rmdir mv6410
[root@localhost ~]# ll
total 96
drwxr-xr-x  3 root root  4096 Feb 26 19:09 Desktop
-rw-----  1 root root  1076 Oct 29 01:16 anaconda-ks
-rw-r--r--  1 root root 55933 Oct 29 01:15 install.log
-rw-r--r--  1 root root  7826 Oct 29 01:14 install.log
-rw-r--r--  1 root root    35 Mar 10 00:43 minicom.log
drwxr-xr-x 11 root root  4096 Mar 11 20:08 speedwee
[root@localhost ~]#

```

폴더 및 파일 완전 삭제

Wrm -r xxx

```

root@localhost:~
[root@localhost ~]# Wrm -r mv6410
[root@localhost ~]#

```

파일 복사

cp 복사할파일 복사될위치

```

root@localhost:~/mv6410
[root@localhost mv6410]# ll
total 4
-rwxrw-r-- 1 root root 528 Mar  5 00:11 bash_profile
[root@localhost mv6410]# cp bash_profile /root/speedwee/
[root@localhost mv6410]#

```

PIPE 명령어 사용 (|)

PIPE 란 앞의 결과 값을 다른 명령어로 인수전달 되는 명령어이다.

ls -l /bin | more, # ps -aux | more, # ps -ef | more

```

root@localhost:~/mv6410
[root@localhost mv6410]# ls -l /bin | more

```

메모리정보

free

```

root@localhost:~/mv6410
[root@localhost mv6410]# free

```

	total	used	free
Mem:	962260	593208	369052
-/+ buffers/cache:	310884		651376
Swap:	1052248	128	1052120

CPU 점유율 정보

top

```

root@localhost:~/mv6410
[root@localhost mv6410]# top

```

top - 22:50:18 up 21 days, 21:47, 3 users
Tasks: 151 total, 2 running, 147 sleeping
Cpu(s): 0.0%us, 0.0%sy, 0.0%ni,100.0%id
Mem: 962260k total, 593516k used, 368744k free
Swap: 1052248k total, 128k used, 1052120k free

PID	USER	PR	NI	VIRT	RES	SHR	S
1	root	15	0	2052	644	548	S
2	root	RT	0	0	0	0	S
3	root	34	19	0	0	0	S
4	root	RT	0	0	0	0	S

환경변수 출력

env | more

```

root@localhost:~/mv6410
[root@localhost mv6410]# env | more
HOSTNAME=localhost.localdomain
SHELL=/bin/bash
TERM=xterm
HISTSIZE=1000
KDE_NO_IPV6=1
QTDIR=/usr/lib/qt-3.3
QTINC=/usr/lib/qt-3.3/include
USER=root
LD_LIBRARY_PATH=/usr/local/arm/4.3.1-eabi-armv6/

```

특정 환경 변수 출력

echo 환경변수이름

```

root@localhost:~/mv6410
[root@localhost mv6410]# echo $PATH
/usr/lib/qt-3.3/bin:/usr/kerberos/sbin:
cal/bin:/sbin:/bin:/usr/sbin:/usr/bin:/
usr/bin/
[root@localhost mv6410]#

```

현재 마운트된 정보 출력

df -h

```

root@localhost:~/mv6410
[root@localhost mv6410]# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/hda1       144G   17G  120G  12% /
tmpfs           471M    0  471M   0% /dev/shm
/root/speedwee/mv6410/rootfs_mv6410.cramfs
46M    46M    0  100% /root/speedwee/mv6410/mv6410_mnt

```

IP 출력

ifconfig

```

root@localhost:~/mv6410
[root@localhost mv6410]# ifconfig
eth0      Link encap:Ethernet  HWaddr 0
inet addr:192.168.0.177  Bcast

```

RPM 설치 및 관리

RPM (RedHat Package Manager)은 리눅스에서 제공되는 기본 패키지이다. 예전에 리눅스는 모든 패키지를 tar을 일일이 설치해야만 했기 때문에 리눅스를 사용하는데 많은 번거로움이 있었다. 그래서 패키지 설치와 관리를 쉽게 하고자 rpm 을 만들게 되었다.

RPM 설치

```
# rpm -i 패키지이름
```

RPM 업데이트

```
# rpm -uvh 패키지이름
```

RPM 패키지정보

```
# rpm -qip 패키지이름
```

RPM 강제설치

```
# rpm -ivh 패키지이름
```

RPM 삭제

```
# rpm -e 패키지이름
```

RPM 설치 전 확인

```
# rpm -qip 패키지이름
```

사용자 계정 관리

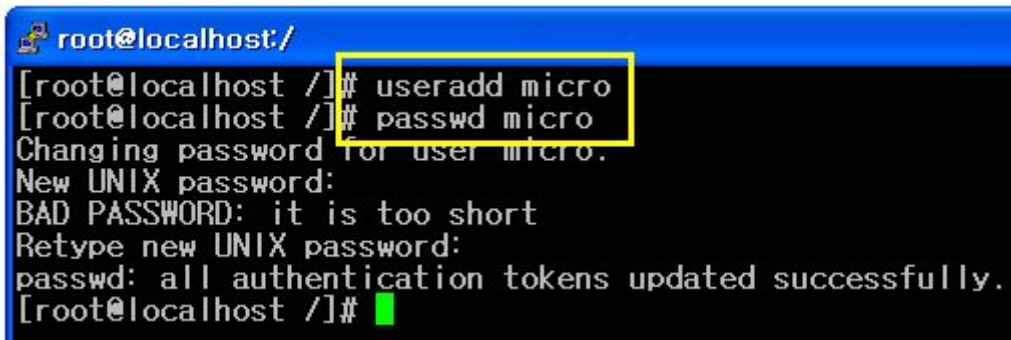
새로운 사용자 계정 만들기 (반드시 ROOT 권한에서 작업해야 한다.)

새로운 계정 생성

```
# useradd xxx
```

```
# passwd
```

비밀번호 입력하면 된다.

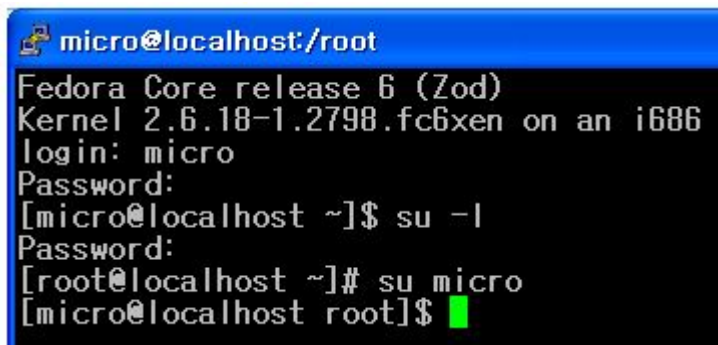


```

root@localhost:/
[root@localhost /]# useradd micro
[root@localhost /]# passwd micro
Changing password for user micro.
New UNIX password:
BAD PASSWORD: it is too short
Retype new UNIX password:
passwd: all authentication tokens updated successfully.
[root@localhost /]# █

```

계정 로그인 방법은 밑에 그림처럼 root 계정으로 가기 위해서는 “# su -” 명령어를 이용하면 된다. 그리고 다시 유저 계정으로 가고 싶으면 “# su 유저계정” 하면된다.



```

micro@localhost:/root
Fedora Core release 6 (Zod)
Kernel 2.6.18-1.2798.fc6xen on an i686
login: micro
Password:
[micro@localhost ~]$ su -l
Password:
[root@localhost ~]# su micro
[micro@localhost root]$ █

```

tar압축, 해제 방법

tar 압축하기

tar cvf 압축파일명.tar 압축할파일명

```
root@localhost:~/mv6410
[root@localhost mv6410]# tar cvf bash_profile.tar bash_profile
bash_profile
[root@localhost mv6410]#
```

tar 압축해제

tar xvf 파일명.tar

```
root@localhost:~/mv6410
[root@localhost mv6410]# tar xvf bash_profile.tar
bash_profile
[root@localhost mv6410]#
```

3. MV6410 Cram File System 이미지 만들기

mkdir mv6410_org <-폴더 생성

mount -o loop rootfs_mv6410.cramfs mv6410_org <-이미지를 폴더에 마운트

tar cvf mv6410_new.tar mv6410_org <- 마운트된 폴더를 tar 로 압축한다.

```
mkdir mv6410_org
mount -o loop rootfs_mv6410.cramfs mv6410_org
tar cvf mv6410_new.tar mv6410_org/
```

umount mv6410_org

tar xvf mv6410_new.tar <- 압축 해제한다

```
umount mv6410_org/
tar xvf mv6410_new.tar
```

mkfs.cramfs mv6410_org mv6410_new <- 압축 해제한 폴더를 이미지로 만들기

```
[root@localhost mv6410_FS]# ll
total 70404
-rw-r--r-- 1 root root 51251200 Mar 19 01:21 mv6410_new.tar
drwxr-xr-x 17 root root 4096 Jan 1 1970 mv6410_org
-rwxr--r-- 1 root root 20754432 Dec 19 13:31 rootfs_mv6410.cramfs
[root@localhost mv6410_FS]# mkfs.cramfs mv6410_org mv6410_new
[root@localhost mv6410_FS]# ls
mv6410_new mv6410_new.tar mv6410_org rootfs_mv6410.cramfs
[root@localhost mv6410_FS]#
```

위에 있는 mkfs.cramfs 명령어는 페도라6 이고 일반리눅스 명령어는 mkfs.cramfs 아닌 cramfs 을 사용한다.

4. MV6410 SD-CARD 마운트

카드 삽입 후 아래와 같이 명령어를 입력한다.

```
# mount /dev/mmcblk0p1 /mnt
```

```
[root@glibc root]# s3c-hsmmc: card inserted.
mmc0: host does not support reading read-only switch.
mmc: major= 254, minor= 0
mmcblk0: mmc0:b368 SM1SD 999936KiB
  mmcblk0: p1

[root@glibc root]# mount /dev/mmcblk0p1 /mnt
[root@glibc root]# cd /mnt
[root@glibc mnt]# ls
608x456.avi*  ins*          mv2443/      racing.avi*
624x352.avi*  lgmm11~1.exe* mv320/       readme.txt*
720x480k.wmv* m2core/      mv6410/     rl_xq_~1.avi
ever.avi*     mfcce400.dll* osdesi~1.sln* run*
image/        mplayer*     player*      s3c_mfc.ko*
[root@glibc mnt]#
```

5. MV6410 Linux Kernel 디렉토리 구조



block : 시스템 타임에 따른 I/O 스케줄러 관련된 폴더

crypto : 암호와 코드에 관련된 폴더

Documentation : 커널에 관련된 각종 텍스트 문서

fs : 가상 파일시스템 등 여러 파일시스템 관련 폴더

init : 리눅스 init 에 관련된 폴더

ipc : 32bit IPC (Inter Process Communication) 세마포어, 메시지큐 등, 관련 폴더

kernel : 명령어 수행을 위한 시스템 콜과 각종 mutex 그리고 시그널 제어에 관련된 폴더

lib : 커널의 관련된 라이브러리 함수 집합 폴더

mm : 세크먼트 디스크립터와 논리에서 선형 그리고 물리 메모리로 접근하는 메모리 관련된 폴더

net : 네트워크 관련된 폴더

scripts : 리눅스 명령어 체계 수행을 위한 각종 스크립트 관련된 폴더

security : 보안에 관련된 폴더

sound : 사운드에 관련된 폴더

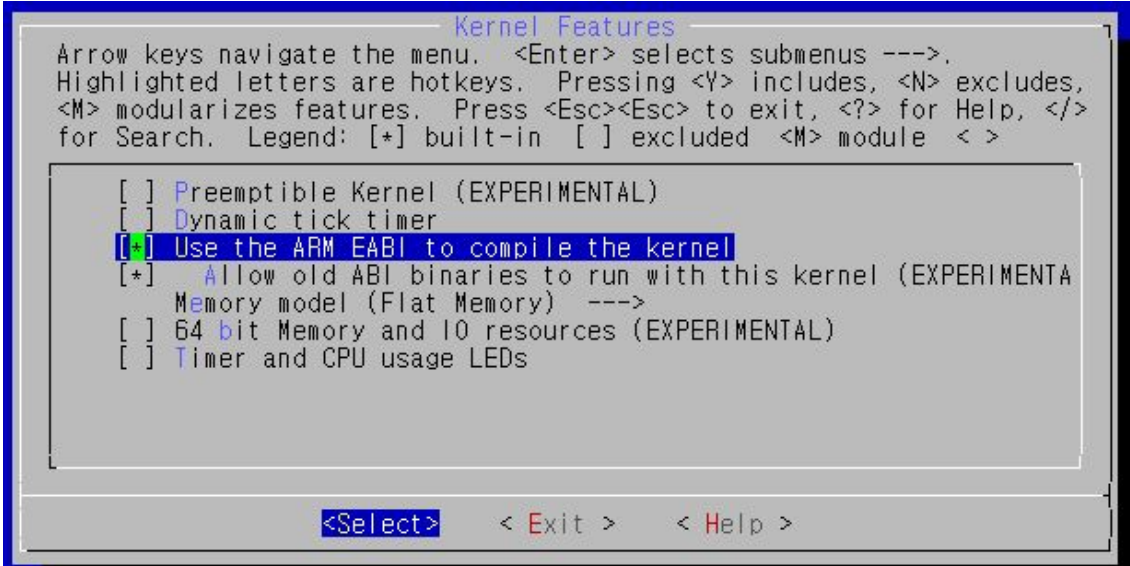
usr : fifo 와 pipe 오브젝트 관련된 폴더

여기서 개발자가 직접 제어할 폴더는 arch, drivers, include 가 있다.

1) arch

이 폴더는 커널에 관계된 아키텍처 코드가 들어 있다.

arch/arm/configs 안에 make menuconfig 명령을 통해 작업을 수행했던 배치 파일이 있다.
“mv6410_defconfig”



또한 make 명령어를 통해 컴파일이 완료 되면 arch/arm/boot 안에 타겟보드에 다운로드하는 “zImage”가 있다.

2) include

이 폴더는 실제 mtd, pmica, network 등 커널에 관련된 각종 헤더 파일이 모여있는 폴더이다.

3) drivers

MV6410-LCD H/W 관련된 각종 드라이버가 있는 소스 부분이다.

- / usb : usb 드라이버
- / serial : UART 드라이버
- / video : LCD 드라이버
- / media : Camera 드라이버
- / net : Ethernet 드라이버

6. 응용프로그램 운영

```
[root@glibc root]# ls
bplay*          fbcam*          rt73.ko         s3c-tvscale.ko
brec*          mpg123*        s3c-tvenc.ko   tv_test*
[root@glibc root]# _
```

bplay : 음성 출력 프로그램

brec : 음성 녹음 프로그램

fbcam : 카메라 프로그램

tv_test : TV OUT 출력 프로그램

1) 음성 프로그램 컴파일 및 실행 방법

(컴파일)



CD 안에 보면 두개의 압축된 파일이 있다.

bplay-0.991.tar.gz : bplay, brec 프로그램

fbcam.tar.gz : Camera 프로그램

다음 명령어를 이용하여 압축을 해제한다.

```
# tar xvf bplay-0.991.tar.gz
```

make

```

root@localhost:~/speedwee/test/bplay-0.991
[root@localhost bplay-0.991]# make clean
rm -f bplay brec *.o *~
[root@localhost bplay-0.991]# make
gcc -Wall -O3 -DUSEBUFFLOCK -c -o bplay.o bplay.c
bplay.c:73: warning: type qualifiers ignored on function
gcc -Wall -O3 -DUSEBUFFLOCK -c -o sndfunc.o sndfunc.c
gcc -Wall -O3 -DUSEBUFFLOCK -c -o shmbuf.o shmbuf.c
shmbuf.c:404: warning: function definition has qualified
gcc bplay.o sndfunc.o shmbuf.o -o bplay
ln -sf bplay brec
[root@localhost bplay-0.991]# ls
COPYING      bplay      bplay.lsm  brec.1      shmbuf.o
Makefile     bplay.l  bplay.o    fmtheaders.h  sndfunc.c
README      bplay.c   brec       shmbuf.c    sndfunc.o
[root@localhost bplay-0.991]#

```

컴파일이 끝나면 bplay, brec 실행 파일이 생성된다.

(실행 방법)

```

[root@glibc mnt]# vi test_

```

위에 그림 처럼 vi test 를 이용해 임의 파일을 만들어 준다.

```

[root@glibc mnt]# ls
ever.avi*      mv6410~/  racing.avi*   s3c_mfc.ko
ins*          nandfl~/  readme.txt*  son.avi*
mplayer*      player*   run*         test*
[root@glibc mnt]# _

```

생성된 "test" 파일

brec 프로그램이 있는 /root 폴더로 이동해 MV6410 메인 보드에 있는 마이크 잭 부분에 마이크를 연결하고 녹음을 한다.

실행 명령어는 다음과 같다.

./brec /mnt/test

```

[root@glibc mnt]# ls
ever.avi*          mv6410~1/      racing.avi*       s3c_mfc
ins*              nandf1~1/     readme.txt*      son.avi
mplayer*         player*        run*             test*
[root@glibc mnt]# cd ..
[root@glibc /]# cd root
[root@glibc root]# ls
bplay*           fbcam*         rt73.ko
brec*            mpg123*       s3c-tvenc.ko
[root@glibc root]# ./brec /mnt/test

```

녹음이 끝나면 “Ctrl + c” 누르면 종료된다.

```

0x68 : 0x0060    <1>0x6A : 0x0000    <1>
0x70 : 0x0000    <1>0x72 : 0x0000    <1>
0x78 : 0x0001    <1>0x7A : 0x0000    <1>

[root@glibc root]# ./bplay /mnt/test_

```

./bplay /mnt/test 수행하면 녹음된 음성을 청취할 수 있다.

2)카메라 프로그램 컴파일 과 실행 방법

(컴파일)

다음과 같은 명령어를 이용해 압축을 해제한다.

```
# tar xvf fbcam.tar.gz
```

```
root@localhost:~/speedwee/test/fbcam
[root@localhost fbcam]# vi Makefile
```

“vi Makefile”로 진입해 밑에 그림 처럼 본인이 압축 해제한 커널 위치를 맞게 바꾸어 준다.

```
root@localhost:~/speedwee/test/fbcam
CC=arm-linux-gcc
CFLAGS=-Wall -I/home/mv6410/kernel/s3c-linux-2.6.21/include
CFLAGS+= -I/home/mv6410/kernel/s3c-linux-2.6.21/drivers/media/video/samsung
all: fbcam
#all: fbcam.old
clean:
rm fbcam -f
```

vi 에디터 종료 후 “make all” 수행하면 “fbcam” 실행 파일 생성된다.

```
root@localhost:~/speedwee/test/fbcam
[root@localhost fbcam]# vi Makefile
[root@localhost fbcam]# make clean
rm fbcam -f
[root@localhost fbcam]# make all
arm-linux-gcc -Wall -I/home/mv6410/kernel/s3c-
0/kernel/s3c-linux-2.6.21/drivers/media/video/s
[root@localhost fbcam]# ls
Makefile fbcam fbcam.c fbcam.tar.gz
[root@localhost fbcam]#
```

(실행 방법)

메인보드에 있는 크레들 컨넥터와 카메라를 연결하고 부팅 후 “./fbcom” 해주면 카메라가 실행된다.

```
[root@glibc root]# ./fbcam
External Camera initialized
Resolution changed back to VGA (640x480) mode (default)
Preview memory required: 0x00320000 bytes
Preview memory required: 0x00258000 bytes
Capturing 640x480x0 "RGBR" images
Images are 0 bytes each
```



3) WIFI 실행 방법

명령어는 다음과 같다.

```
# insmod rt73.ko
# ifconfig rausb0 192.168.xxx.xxx
# iwlist scanning
```

```
[root@glibc root]# usb 1-1: new full speed USB device
ress 6
usb 1-1: configuration #1 chosen from 1 choice

[root@glibc root]# insmod rt73.ko
idVendor = 0x148f, idProduct = 0x2573
usbcore: registered new interface driver rt73
[root@glibc root]# ifconfig rausb0 192.168.0.236
=> usb_rtusb_open
[root@glibc root]# iwlist scanning
```

여기서 AP 장비 이름이 검색 되는데 그 장비 이름을 기입해 주면 된다.

```
# iwconfig rausb0 essid [Write! Ap of name ]
```

```
Channel:6
Encryption key:off
Bit Rates:36 Mb/s
Cell 06 - Address: 08:10:74:03:BA:8A
Mode:Managed
ESSID:"vipsap3"
Channel:7
Encryption key:on
Bit Rates:1 Mb/s
Cell 07 - Address: 00:13:10:3B:BB:37
Mode:Managed
ESSID:""
Channel:7
Encryption key:on
Bit Rates:108 Mb/s
Cell 08 - Address: 00:1F:1F:17:5C:F8
Mode:Managed
ESSID:"CubicAP-1"
Channel:11
Encryption key:on
Bit Rates:0 kb/s

[root@glibc root]# iwconfig rausb0 essid vipsap3
[root@glibc root]# _
```


4) TV-OUT 실행 방법

명령어는 다음과 같다.

```
# insmod s3c-tvscaler.ko
# insmod s3c-tvenc.ko
# ./tv_test 0 -> Composite tv out
# ./tv_test 0 0 -> Composite tv out
# ./tv_test 0 1 -> S-Video tv out
```

```
[root@glibc root]# insmod s3c-tvscaler.ko
S3C TV Scaler  Init : Success
Done
[root@glibc root]# insmod s3c-tvenc.ko
S3C TV Encoder  Init : Success
Done
[root@glibc root]# ./tv_test 0 0
Device file opens3c_tvscaler_init:
TV-OUT: VIDIOC_ENUMINPUT : index = 0
TV-OUT: VIDIOC_S_INPUT
TV-OUT: VIDIOC_ENUMOUTPUT : index = 0
TV-OUT: VIDIOC_S_OUTPUT
C: VIDIOC_S_FMT
P: VIDIOC_S_CTRL

pPreBuff = 0x40001000
pPostBuff = 0x40401000
V4L2 APPL : Name of the interface is S3C TV-OUT driveTVENCODER
V4L2 APPL : [0]: IN channel name LCD FIFO_OUT
V4L2 APPL : LCD FIFO INPUT
V4L2 APPL : [0]: OUT channel name TV-OUT
V4L2 APPL : TV-OUT
```