실험 실습 안내서 (#2)

제 출 자	학 과 : 전자공학과	학 번 :21611668	성 명 :이현정
실험 번호(Number)	Embedded System - 02		
실험 제목(Subject)	리눅스 명령어 사용		
실험 목적(Objectives)	리눅스에는 어떤 명령어들이 있으며, 그 명령어들은 어떤 개념을 가지고 있고, 또한 어떤 작업을 하는지를 알아본다.		
기초 지식 (Background)	리눅스의 명령어의 개념과 종류 및 동작형태		
자 료 (Resource & reading Material)	Intel PXA255와 임베디드 리눅스 응용 저 자: (주) 휴인스 기술연구소 출판사: 홍릉과학 출판사 수업 자료(실험실습 안내서, “2-1장 리눅스 명령어”)		
설비 및 준비사항 (Tools & Preparation)	리눅스를 O.S로 하는 Computer		
전체 절차 (Steps)	① 파일 관리 명령어 실습 (cd, ls, cp, mv, rm, mkdir, rmdir, cat, pwd, chmod ... 등등) ② 프로세스 관리 명령어 실습 (ps, pstree, top, shutdown, reboot, kill ... 등등) ③ 시스템 정보 명령어 실습 (arch, cal, clock, date, df, free, hostname, tty, whereis ... 등등) ④ 사용자 관리 명령어 실습 (su, finger, id, logname, uptime, w, who, whoami, adduser ... 등등) ⑤ 검색 명령어 실습 (grep, find ... 등등) ⑥ 디스크 관리 명령어 실습 (mount/umount, fsck ... 등등) ⑦ NETWORK 명령어 실습 (ping, ifconfig ... 등등) ⑧ vi 에디터 사용법		
주의사항 (Cautions)			

실험 실습 결과보고서

구체적인 절차 기술
(Detail steps)

1. 실습 내용

① 목 적:

- LINUX 시스템의 일반 사용자들을 위한 기본적인 명령에 대하여 알아본다.
- 시스템의 시작/로그인 및 종료/로그아웃.
- 파일 관리/디렉토리 관리.
- 사용자 관리.

☞ 명령어 및 유틸리티 리스트:

cancel	group	paswd
cat	head	pwd
cd	lp	rm
chgrp	lpstat	rmdir
chmod	ls	stty
chown	mail	tail
clear	man	test
cp	mkdir	vi
date	more	
emas	mv	
file	newgrp	

☞ 사용자의 등록:

- 자신의 소속 조직의 리눅스 관리자로부터 사용자 계정을 받는다.
- 시스템관리자 또는 서버 관리자로부터 계정을 부여 받는다.
- 본인의 컴퓨터에 리눅스를 설치하고 직접 관리자 자격으로 리눅스를 사용한다.

☞ 로그인:

- 자신의 사용자이름과 초기의 패스워드는 시스템 관리자로부터 부여받거나, 또는 자신 소유의 리눅스 시스템을 소유한 경우에는 어떤 표준 값으로 설정된다.
- 리눅스는 먼저 "login:"을 로그인 프롬프트로 화면에 보임으로써, 사용자이름을 질문한다. 그런 다음 사용자의 패스워드를 질문한다.
- 리눅스는 영문자의 대소문자를 구별하는 시스템이다.
- 리눅스 시스템에서는 "[사용자이름@Linux 현디렉토리]셸프롬프트"로 표시.

```
Red Hat Linux release 7.3 (Valhalla)
Kernel 2.4.18-3 on an i686
login: user
Password:                          ,,,,,, 여기에 입력된 내용은 표시되지 않는다.
Last login: Wed Aug 28 02:15:10 from 211.54.35.181
[user@Linux user]$ _
```

실험 실습 결과보고서

구체적인 절차 기술
(Detail steps)

☞ 쉘(Shell):

- 셸은 순수 리눅스 운영체제와 사용자 사이에서 중간적인 매개 역할을 하는 프로그램.
- 셸은 사용자가 프로그램을 수행하고, 프로세스들의 파이프라인을 만들고, 출력을 파일에 저장하며, 동시에 하나 이상의 프로그램이 수행되도록 한다.
- 많이 사용되는 리눅스의 셸

  Bourne-Again 셸

  Z 셸

  TC 셸

☞ 유틸리티의 실행:

- 유틸리티를 실행하기 위해서는, 유틸리티 이름을 입력하고 엔터 키를 누른다.
- 엔터 키는 사용자가 리눅스에게 명령어를 입력하였으니, 그것을 실행하라고 지시하는 것이다.
- 모든 시스템은 정확하게 동일한 유틸리티를 갖지 않는다.

☞ 구문 설명을 위한 BNF:

Utility **date** [*yymmddhhmm*[.ss]] [MMDDhhmm[[CC]YY][.ss]]

인수가 없다면, **date**는 현재의 날짜와 시간을 표시한다. 만약 인수가 제공된다면, **date**는 날짜를 주어진 인수에 따라 설정할 수 있게 하는데, 여기서 MM은 월의 숫자, DD는 오늘의 날짜를 나타내는 숫자, hh는 시간을(24시간제로), 그리고 mm은 분을 나타내는 숫자이다. 선택적으로 사용하는 CC는 그해의 첫번째 두 숫자, YY는 그 해의 마지막 두 숫자이고, ss는 초를 나타내는 숫자이다. 단지 슈퍼 유저만이 날짜를 설정할 수 있다.

Utility **clear**

이 유틸리티는 화면을 지운다.

☞ 온라인 도움말 얻기 : man

모든 리눅스 시스템은 man(manual pages의 약어)이라는 유틸리티를 가지는데, 이 유틸리티는 키보드를 눌러주는 것만으로 유틸리티 사용에 관한 정보를 보여준다.

Utility **man** [-S section_list] word
man -k keyword

매뉴얼 페이지(manual pages)는 원본의 리눅스 문서에 대한 온라인 버전으로, 리눅스 문서는 유틸리티, 시스템 호출(system call), 파일 형식, 그리고 셸 등에 관한 정보를 포함한다. **man**이 주어진 유틸리티에 대한 도움말을 보여줄 때, 그 내용이 어느 장으로부터 발췌되는지를 표시해준다.
man의 첫번째 형식은 word와 연관된 매뉴얼 내용을 보여준다. 만약 장(section) 번호가 명시되지 않는다면, 그 word가 발견되는 첫번째 내용을 보여준다. **man**의 두번째 형식은 keyword를 포함하는 모든 매뉴얼 내용들을 보여준다.

실험 실습 결과보고서

구체적인 절차 기술
(Detail steps)

☞ 특수 문자:

- 일부 문자들은 리눅스 터미널에 입력되었을 때 특수하게 해석된다. 이러한 문자들은 때때로 메타 문자라 호칭된다.
- 옵션 -a(all)을 갖는 stty 유틸리티를 이용하여 이 특수 문자들을 모두 표시할 수 있다.

옵션	의미
erase	한 문자를 지움(backspace)
kill	현재 작업중인 줄을 모두 지움
werase	마지막 단어를 지움
rprnt	그 줄을 다시 출력함
flush	대기중인 입력을 무시하고 그 줄을 다시 출력함
lnext	다음 문자를 특수 문자로 처리하지 않음
susp	추후 수행하기 위해 프로세스를 일시 정지시킴
intr	코어 덤프(core dump) 없이 전면 작업을 종료함
quit	코어 덤프하고 전면 작업을 종료함
stop	터미널 출력을 정지/다시 시작함
eof	입력의 끝
eol	줄의 끝

☞ 중요 특수 문자:

- 프로세스의 종료: Control-C
- 출력의 정지: Control-S/Control-Q
- 입력의 끝: Control-D

☞ 패스워드 설정 : passwd

- 최소 6문자 이상, 사전상의 단어나 고유 명사는 피한다.

Utility **passwd**

passwd는 사용자의 패스워드를 바꿀 수 있도록 해준다. 사용자는 이전 패스워드를 입력하도록 요구받고, 그런 다음 새로운 패스워드를 두 번 입력하도록 요구받는다. (입력한 내용이 화면에 보이지 않으므로 입력 오류를 모르게 된다. 만약 새로운 패스워드의 두 번의 입력이 일치하지 않으면 두 번 중의 한번은 입력 오류이므로, 새로운 패스워드를 다시 입력하도록 요청된다. 새로운 패스워드는 패스워드 파일인 "/etc/passwd" 또는 "shadow"(더 나은 보안을 위하여) 파일에 암호화된 형태로 저장된다.

```
[user@Linux user]$ passwd
Changing passwd for user
<current> UNIX password: [illegible]
New UNIX password: [illegible]
Retype new UNIX password: [illegible]
passwd: all authentication tokens updates successfully
[user@Linux user]$ _
```

☞ 로그 아웃/시스템 종료/재부팅:

- 접속 끊기 및 로그아웃: Control-D, logout, exit
- 시스템 종료 : shutdown, halt
- 시스템 재시작 : reboot

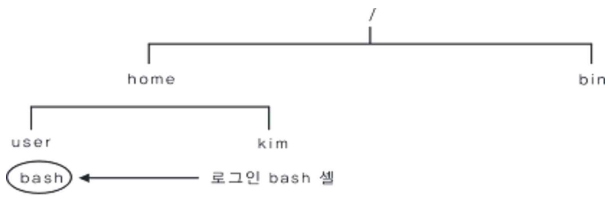
☞ 현재 작업중인 디렉토리의 확인 : pwd

- 모든 사용자는 각기 다른 자신의 사용자 홈 디렉토리를 가지며, 이것은 보통 "/home"로 시작된다.

Utility **pwd**

현재의 작업 디렉토리가 무엇인지 출력한다.

- 로그인 쉘

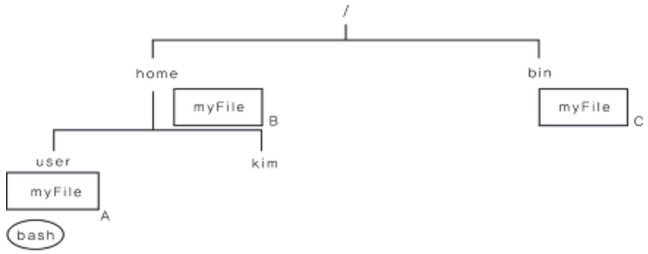


실험 실습 결과보고서

구체적인 절차 기술
(Detail steps)

☞ 절대경로 이름과 상대경로 이름:

- 동일한 이름을 갖는 동일한 여러 개의 파일들이 여러 다른 디렉토리들에 존재하는 것은 가능하나, 동일한 디렉토리에 동일한 이름을 갖는 2개의 파일은 존재할 수 없다.



☞ 절대경로 이름과 상대경로 이름:

- 절대경로 이름

파일	절대 경로 이름
A	/home/user/myFile
B	/home/myFile
C	/bin/myFile

- 상대경로 이름 : /home/user 기준

필드	의미
.	현재 디렉토리
..	부모 디렉토리

필드	의미
A	myFile
B	../myFile
C	../../bin/myFile

☞ 파일의 생성 : cat

Utility **cat** –n { fileName }*

cat 유틸리티는 표준 입력이나 파일로부터 입력받고, 이를 표준 출력에 표시한다. –n 옵션은 출력에 줄 번호를 덧붙인다. **cat**은 "catenate"의 축약어이고, 그 의미는 "연결하여 붙인다"는 의미이다.

- 프로세스의 표준 입력은 키보드이고, 표준 출력은 화면이다.
- 사용자는 출력 리다이렉션 이라는 셸의 기능을 이용하여 프로세스의 표준 출력을 화면 대신에 파일로 보낼 수 있다.
- 명령 다음에 '>' 문자와 '파일 이름'이 뒤따른다면, 그 명령으로부터의 출력은 파일에 저장된다.
- 파일이 존재하지 않는다면 새로이 생성되고, 그렇지 않다면 파일에 이미 들어 있던 내용 위에 덮어서 쓰여진다.

```
[user@Linux user]$ cat > Queen        ...키보드 입력을 "Queen" 파일에 저장
Don't make no difference
If I'm wrong or I'm right,
I've get the feeling
And I'm willing tonight.
Well, I ain't anybody's angel.
What can I say?
Well, I'm only that ways.
^D                                    ...cat에게 입력의 끝에 도달했음을 알림
[user@Linux user]$ _
```

실험 실습 결과보고서

구체적인 절차 기술
(Detail steps)

☞ 파일의 내용보기 : more

Utility: **more** –f { *+lineNumber* }{ *fileName* }*

more 유틸리티는 파일들의 리스트를 한 번에 한 페이지씩 표시할 수 있게 한다. 디폴트로, 각 파일은 줄 1에서부터 표시되며, + 옵션은 시작 번호를 명시하기 위해 사용된다. –f 옵션은 긴 줄을 연속적으로 나타낸다. 각 페이지가 출력된 후에, **more**는 명령을 기다리고 있다는 의미로서 "--More--"라는 메시지를 표시한다. 다음 페이지를 출력하기 위해서는 스페이스바 키(spacebar key)를 누르면 된다. 다음 줄을 출력하기 위해서는 엔터 키를 누르면 된다. **more**를 빠져 나오기 위해서는 "**q**" 키를 치면 된다. 다른 명령들에 대한 도움말을 얻기 위해서는 "**h**" 키를 치면 된다.

☞ 파일의 내용보기 : head/tail

Utility: **head** –n {*fileName*}*

head 유틸리티는 파일의 처음 n줄을 표시한다. 만일 n이 명시되지 않으면, n은 10으로 간주된다. 만일 두 개 이상의 파일을 지정하면, 각 파일의 내용을 보여주기 전에 각 파일을 확인할 수 있는 정보를 먼저 보여준다.

Utility: **tail** –n {fileName}*

tail 유틸리티는 파일의 마지막 n개의 줄을 화면에 보여준다. 만일 n이 명시되지 않으면, n은 10으로 간주된다. 만일 두개 이상의 파일을 지정하면, 각 파일의 내용을 보여주기 전에 각 파일을 확인할 수 있는 정보를 먼저 보여준다.

☞ 파일의 이름 바꾸기 : mv

Utility: **mv** –i *oldFileName newFileName*
mv –i {*fileName*}* *directoryName*
mv –i *oldDirectoryName newDirectoryName*

mv의 첫번째 형식은 oldFileName을 newFileName으로 바꾼다. 만일 newFileName이 이미 존재한다면, 그 파일의 내용은 oldFileName의 내용으로 바뀐다. 두번째 형식은 파일을 다른 디렉토리로 이동시킬 수 있게 하며, 세번째 형식은 전체 디렉토리를 이동시킬 수 있게 한다. 만약 목적지 위치가 원래의 위치와 동일한 파일 시스템에 있다면 이러한 옵션들은 실제 파일의 내용들을 이동하는 것이 아니라, 대신에 계층적 구조에서 이름만 이동시킨다. 따라서 **mv**는 매우 빠른 유틸리티이다. –i 옵션은 의도하지 않은 내용의 변경을 방지하기 위해, newFileName이 이미 존재하는 경우에는 확인을 위한 프롬프트를 보여준다.

☞ 디렉토리 만들기 : mkdir

Utility: **mkdir** [–p] *newDirectoryName*

mkdir 유틸리티는 디렉토리를 생성한다. –p 옵션은 존재하지 않는 newDirectoryName의 경로 이름에 부모 디렉토리를 생성한다. 만약 newDirectoryName이 이미 존재한다면 에러 메시지가 표시되고, 존재하는 파일은 어떤 변화도 없다.

```
[user@Linux user]$ mkdir Houston            ...디렉토리 "Houston"을 생성
[user@Linux user]$ ls -lF                   ...새로운 디렉토리가 생성되었는지 확인
total 5
drwxr-xr-x   2 user  syslab  112  Aug 28  02:29 Desktop/
-rw-rw-r--   1 user  syslab  163  Sep  4  03:43 Queen.perl
drwxrwxr-x   2 user  syslab   48  Sep  4  10:07 Houston/
[user@Linux user]$ _
```

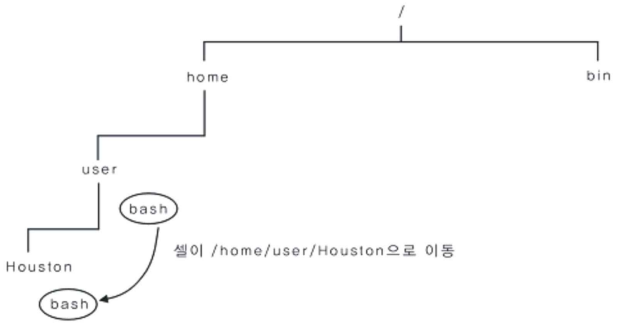
실험 실습 결과보고서

구체적인 절차 기술
(Detail steps)

☞ 다른 디렉토리로 이동 : cd

Shell Command **cd** [*directoryName*]

cd 셸 명령은 셸의 현재 작업중인 디렉토리를 지정한 이름의 디렉토리로 변경한다. directoryName의 인수가 없는 경우에, 셸은 사용자의 홈 디렉토리로 옮겨진다.



☞ 파일 복사 : cp

Utility: **cp** –i *oldFileName newFileName*
cp –iR {*fileName*}+ *directoryName*

cp의 첫번째 형식은 oldFileName의 내용을 newFileName으로 복사한다. newFileName이 이미 존재한다면,그 내용은 oldFileName의 내용에 의해 대체된다. **–i** 옵션은 우연히 기존 내용에 덮어쓰지 않도록 newFileName이 이미 존재한다면 확인을 위한 절차를 거친다. **cp**의 두번째 형식은 directoryName을 갖는 디렉토리로 파일을 복사한다. **–R** 옵션은 디렉토리인 소스 파일을 재귀적으로 복사하여, 전체 디렉토리 구조를 복사할 수 있다.

cp가 수행하는 일

원래 파일의 내용을 물리적으로 복사

계층적 디렉토리 구조 안에서 복사된 파일을 지시하는 새로운 레이블 생성

☞ 디렉토리 제거 : rmdir

Utility: **rmdir** {*directoryName*}+

rmdir 유틸리티는 명령어와 함께 제공되는 디렉토리 이름들의 목록에 있는 디렉토리 모두를 제거한다. 디렉토리가 제거되기 전에, 그 디렉토리에 포함된 내용들은 모두 미리 제거되어야만 한다. 디렉토리와 그 내용 모두를 재귀적으로 제거하기 위해서는 다음절에서 설명되는 **–r** 옵션으로 **rm** 유틸리티를 사용하라.

디렉토리내에 파일이 존재하면 에러가 생긴다.

☞ 파일의 삭제 : rm

Utility: **rm** –fir {*fileName*}+

rm 유틸리티는 계층적 디렉토리로부터 파일의 레이블을 삭제한다. 만일 그 fileName이 존재하지 않는다면, 에러 메시지를 보여준다. **–i** 옵션은 파일을 삭제하기 전에 사용자에게 확인을 요구한다. **y**를 누르면 삭제를 확인하는 것이고, 파일 삭제를 취소하는 경우에는 **n**을 누르면 된다. fileName이 디렉토리 이름인 경우 **–r** 옵션을 사용함으로써 서브디렉토리들을 포함한 모든 내용을 재귀적으로 지운다. **–f** 옵션은 어떤 에러 메시지나 프롬프트도 나타내지 않는다.

rm 유틸리티는 계층 구조로부터 파일의 레이블을 삭제한다. 더 이상 파일을 참조하는 레이블이 없다면, 리눅스는 그 파일 자체를 삭제한다.

실험 실습 결과보고서

구체적인 절차 기술
(Detail steps)

☞ 파일의 프린팅 : lp/lpstat/cancel

Utility: **lp** [*-d destination*] [*-n copies*] {*fileName*}*

lp는 **-d** 옵션에 의해 지정된 프린터로 명시된 파일을 출력한다. 만일 파일이 명시되지 않으면, 대신 표준 입력을 출력한다. 디폴트로서 각 파일은 한 번씩 출력된다. 그러나, 복사물의 수를 나타내는 **-n** 옵션을 이용하여 재설정할 수 있다.

Utility: **lpstat** [destination]

lpstat는 **lp** 명령어로 프린터로 전송된 모든 프린터 작업들의 상태를 보여준다. 만약 목적프린터가 명시되어 있으면, **lpstat**는 그 프린터에 대한 대기 정보만을 보여준다. **lpstat**는 사용자, 작업의 이름 및 크기, 프린터-요청 ID를 보여준다.

Utility: **cancel** [request-ID]+

cancel은 프린터 큐로부터 명시된 모든 작업을 취소한다. 만일 사용자가 수퍼 유저라면, 다른 사람에 의해 요청된 작업을 포함하여 모든 작업을 취소할 수 있다.

☞ 파일 내의 단어 수 세기 : wc

Utility: **wc** -lwc {*fileName*}*

wc 유틸리티는 파일들의 목록 내의 문자, 단어, 줄의 수를 센다. **-l** 옵션은 줄의 수만을, **-w** 옵션은 단어의 수만을 그리고 **-c** 옵션은 문자의 수만을 센다. 만일 옵션이 주어지지 않는다면, 위 세 가지 모두를 표시한다. 단어는 탭(tab), 공백(space), 그리고 새 줄(newline)에 둘러싸인 경우로 정의한다.

☞ 파일 속성

```
[user@Linux Houston.final]$ ls -lsF Queen.final
 4 -rw-rw-r--   1 user  syslab 343 Sep  4 10:44 Queen.final
[user@Linux Houston.final]$ _
```

필드 번호	필드 값	의미
1	4	파일에 의해 점유된 저장 블록 수
2	-rw-rw-r--	파일을 읽고, 쓰고, 실행할 수 있는 사람을 나타내는 파일의 허가권 모드
3	1	하드 링크 계수(본서의 뒤에서 논의된다)
4	user	파일 소유자의 사용자 ID
5	syslab	파일의 그룹 ID
6	343	바이트 단위로서 파일의 크기
7	Sep 4 10:44	파일이 마지막으로 변경된 시간
8	Queen.final	파일 이름

- 필드 1 : 파일 저장 공간

  파일이 디스크 공간을 실제적으로 얼마나 점유하는지 알고자 할 때 유용하다.

- 필드 2 : 파일의 유형과 허가권

  첫번째 문자는 부호화된 파일의 유형을 나타낸다

문자	파일 유형
-	정규 파일
d	디렉토리 파일
b	(디스크 드라이브와 같이) 버퍼된 특수 파일
c	(터미널 같이) 버퍼되지 않은 특수 파일
l	심볼릭 링크
p	파이프
s	소켓

실험 실습 결과보고서

구체적인 절차 기술
(Detail steps)

☞ 파일 속성:

- 필드 2 : 파일의 유형과 허가권

  파일의 유형 확인 : file.

Utility: **file** *{fileName}+*

file 유틸리티는 파일의 내용이 쓰여진 언어를 포함하여 인수 fileName의 내용을 묘사한다. 심볼릭 링크 파일에 **file** 유틸리티를 사용한 경우, 연결 자체보다는 연결에 의해 지적되는 파일에 대한 정보를 보여준다.

```
[user@Linux Houston.final]$ file Queen.final
Queen.final: ASCII English text
[user@Linux Houston.final]$ _
```

사용자(소유주)	그룹	다른 사용자
rw-	rw-	r--

읽기 허가	쓰기 허가	실행 허가
r	w	X

	정규 파일	디렉토리 파일	특수 파일
읽기	프로세스는 그 내용을 읽을 수 있다.	프로세스는 디렉토리를 읽을 수 있다. (즉 디렉토리가 포함하고 있는 파일의 이름을 표시한다)	프로세스는 read( ) 호출을 이용하여 파일을 읽을 수 있다.
쓰기	프로세스는 내용을 변화시킬 수 있다.	프로세스는 디렉토리로부터 파일을 제거하거나 디렉토리로 덧붙일 수 있다.	프로세스는 write( ) 호출을 이용하여 파일에 쓸 수 있다.
실행	프로세스는 파일을 실행할 수 있다. (단지 그 파일이 프로그램인 경우에 한하여)	프로세스는 디렉토리에 또는 디렉토리의 서브 디렉토리 중 하나에 포함되어 있는 파일에 접근할 수 있다.	아무 의미가 없다.

- 프로세스가 갖는 파일 허가권과 관련한 4가지 값

  실제 사용자 ID(real user ID)

  유효 사용자 ID(effective user ID)

  실제 그룹 ID(real group ID)

  유효 그룹 ID(effective group ID)

- 프로세스가 실행될 때 파일 허가권의 적용

  프로세스의 유효 사용자 ID와 파일의 소유자가 같다면, 사용자의 허가권이 적용된다.

  프로세스의 유효 사용자 ID가 파일의 소유자와는 다르지만, 프로세스의 유효 그룹 ID가 파일의 그룹 ID와 같다면, 그룹의 허가권이 적용된다.

  프로세스의 유효 사용자 ID나 프로세스의 유효 그룹 ID가 그 파일의 소유자 및 그 파일의 그룹 ID와 각각 일치하지 않으면, 다른 사용자의 허가권이 적용된다.

- set user ID/set group ID

  set user ID 허가권을 가진 실행 파일이 실행될 때, 프로세스의 유효 사용자 ID는 실행 파일의 유효 사용자 ID가 된다.

  set group ID 허가권을 가진 실행 파일이 실행될 때, 프로세스의 유효 그룹 ID는 실행 파일로부터 복사된다.

  이 두 가지 경우에서, 실제 사용자 및 그룹 ID는 어떤 영향도 받지 않는다.

- 파일 허가권에 관련된 참고 사항

  프로세스가 파일을 만들 때, 그 파일에 주어진 허가권은 umask라 불리는 특별한 값에 의해서 수정된다.

  수퍼 유저는 자동적으로 모든 접근 권리를 가진다.

  파일의 소유자가 그룹 또는 다른 사용자보다 적은 허가권을 가질 수도 있다.

- 필드 3 : 하드 링크 개수

  필드 3은 계층 내에 얼마나 많은 레이블이 동일한 물리적 파일을 가리키는가를 지시하는 파일의 하드 링크 수를 보여준다.

실험 실습 결과보고서

구체적인 절차 기술
(Detail steps)

- 필드 4 : 파일의 소유자
  모든 리눅스 프로세스는 소유주를 가지며, 그 소유주는 전형적으로 그 프로세스를 시작한 사용자의 사용자이름과 동일하다.
  프로세스가 파일을 생성할 때마다 그 파일의 소유주는 그 프로세스의 소유주로 설정된다.
  사용자 이름으로 알려진 문자열은 전형적으로 사용자를 나타내는 방법이지만, 리눅스는 내부적으로는 사용자 ID라고 알려진 정수로 표현한다.
- 필드 5 : 파일이 그룹
  프로세스에 의해 생성된 파일은 그 프로세스를 생성한 사용자와 동일한 그룹에 할당.
  그룹도 이름의 문자열에 의해 참조되지만 내부적으로는 그룹 ID라 불리는 정수 값으로 표현된다.
- 필드 6 : 바이트 단위로서 파일의 크기
  실제 파일의 크기를 바이트 단위로 표현한다.
- 필드 7 : 파일의 마지막 변경 시간
  make 유틸리티는 파일의 마지막 변경 시간을 파일간의 상호 관계 검사를 제어하기 위해 사용한다.
  find 유틸리티는 파일의 마지막 변경 시간을 기초로 하여 파일을 찾도록 프로그램될 수 있다.
- 필드 8 : 파일의 이름
  리눅스의 파일 이름은 255자 길이까지 가능하다.
  /를 제외한 어떤 프린트 가능한 문자로도 구성이 가능하다.
  <, >, *, ?, tab 과 같이 사용자 및 셸 모두에게 혼란을 일으킬 수 있는 특수 문자의 사용은 피하도록 권한다.
  리눅스 시스템에서는 확장자를 파일 이름 끝에 요구하지는 않는다.
  사용자가 정의하여 선택할 수 없는 유일한 파일 이름은 "."과 ".."이다.

☞ 그룹 목록 보기 : group

Utility: **chgrp** –R *groupname* {*fileName*}+

chgrp 유틸리티는 자신 소유 파일의 그룹을 변경할 수 있게 한다. 수퍼 유저는 임의의 파일 그룹을 변경할 수 있다. 인수 groupname의 뒤에 위치하는 모든 파일이 영향을 받는다. **–R** 옵션은 디렉토리 내의 파일의 그룹을 재귀적으로 변경한다.

☞ 파일 그룹 바꾸기 : chgrp

Utility: **groups** [*userId*]

인수 없이 호출될 때, **groups** 유틸리티는 사용자가 회원으로 속해 있는 모든 그룹의 목록을 표시한다. 사용자의 이름이 명시되어 있다면, 사용자 그룹의 목록이 표시된다.

☞ 파일의 허가권 바꾸기 : chmod

Utility: **chmod** –R *change* {, *change*}+{*fileName*}+

chmod 유틸리티는 인수 change에 따라 명시된 파일의 모드를 변경한다. 인수 change는 다음과 같은 형식을 취한다.

clusterSelection+newPermissions (허가권 추가)
clusterSelection–newPermissions (허가권 제거)
clusterSelection=newPermissions (절대적 허가권 할당)

여기서 clusterSelection은 아래의 기호들의 임의의 조합이다.

- u(user/owner)
- g(group)
- o(others)
- a(all)

또한 newPermissions는 아래의 기호들의 임의의 조합이다.

- r(read)
- w(write)
- x(execute)
- s(set user ID/set group ID)

–R 옵션은 디렉토리 내의 파일의 모드를 재귀적으로 변경시킨다. 디렉토리의 허가권 설정 변경은 그 디렉토리에 포함된 파일의 허가권을 변경시키지 않는다.

실험 실습 결과보고서

구체적인 절차 기술
(Detail steps)

☞ 파일의 허가권 바꾸기 : chmod

요구 사항	허가권 변경
그룹 쓰기 허가권 추가	g+w
사용자 읽기 및 쓰기 허가권 제거	u-rw
사용자, 그룹, 일반인에 대한 실행 허가권 추가	a+x
그룹에 읽기 허가권만을 주기	g=r
사용자에 쓰기 허가권 추가, 그리고 그룹의 읽기 허가권 제거	u+w, g-r

	사용자	그룹	모든 사용자
설정	rwx	r-x	---
2진수	111	101	000
8진수	7	5	0

- 표준 파일 퍼미션:

644 : 슈퍼유저의 파일

664 : 일반 사용자의 파일

755 : 슈퍼유저의 실행 파일

775 : 일반 사용자의 실행 파일

755 : 슈퍼유저의 디렉토리

775 : 일반사용자의 디렉토리

☞ 파일의 소유권 바꾸기 : chown

Utility: **chown** −R *newUserId {fileName}+*

수퍼 유저는 **chown** 유틸리티를 이용하여 파일의 소유권을 변경할 수 있다. newUserId의 다음에 위치한 모든 파일은 영향을 받는다. **−R** 옵션은 재귀적으로 디렉토리 내의 파일에 대한 소유권을 변경한다.

☞ 그룹의 변경 : newgrp

Utility: **newgrp** [−] [groupname]

newgrp 유틸리티는 그룹 이름이 인수로서 불리워질 때, 그룹 이름에 해당하는 유효 그룹 ID를 갖는 새로운 셸을 생성한다. 기존의 셸은 새롭게 생성된 셸을 끝내기 전까지는 잠든다. 사용자는 자신이 명시한 그룹의 회원이 되어야만 한다. 만일 그룹 이름 대신에 대시(−)를 인수로 사용한다면, 시스템에 로그인할 때 생성되었던 셸과 동일한 설정을 갖는 셸이 생성된다.

☞ 터미널의 특성 변경 : stty

Utility: **stty** −a *{option}*{metacharacterString <value>}**

stty 유틸리티는 터미널의 특성을 설정하거나 조사하는데 사용한다. **stty**는 100가지 이상의 다른 설정으로 변경할 수 있게 한다. 여기서는 가장 일반적인 것들만을 보인다. 좀더 자세한 것은 **man**을 이용하라. 현재의 터미널 설정 상태를 보기 위해서는 **−a** 옵션을 사용하면 된다. 특별한 설정을 위해서는 다음과 같은 옵션 중 하나 이상을 사용한다.

옵션	**의미**
−echo	타이핑한 문자들을 다시 보여주지 않는다.
echo	타이핑한 문자들을 다시 보여준다.
−raw	메타문자의 특별한 의미를 사용할 수 있도록 한다.
raw	메타문자의 특별한 의미를 사용할 수 없도록 한다.
−tostop	후면 작업이 터미널에 출력을 보낼 수 있도록 한다.
tostop	후면 작업이 터미널에 출력을 보내는 것을 중단한다.
sane	터미널 특성들을 사전에 정한 값으로 설정한다.

실험 실습 결과보고서

구체적인 절차 기술
(Detail steps)

메타문자의 이름에 대응하는 문자열 다음에 새로운 값을 넣어줌으로써 메타문자의 연결관계를 설정할 수 있다. 제어문자는 ^를 문자 앞에 위치시키거나 실제 제어문자 앞에 \를 타이핑함으로써 나타낼 수 있다. 다음은 일반적인 메타문자열과 그 의미이다.

옵션	의미
erase	한 문자를 지움
kill	현재 작업중인 줄을 모두 지움
lnext	다음 문자를 특수문자로 처리하지 않음
susp	나중에 깨우도록 하고, 프로세스를 임시 정지
intr	코어 덤프 없는 전면 작업 종료(인터럽트)
quit	코어 덤프 있는 전면 작업 종료
stop	터미널 출력을 정지/다시 시작함
eof	입력의 끝(또는 파일의 끝)

☞ vi 편집기의 시작

Bill Joy에 의해 BSD 유닉스용으로 개발

System V와 유닉스의 대부분 버전에서 기본 유틸리티로 채택

vi는 visual editor의 약어이다.

처음 vi를 시작하려면, 아무런 인수 없이 vi라고 타이핑하면 된다. 존재하는 파일을 편집하기 위해서는 명령 줄에 인수로서 그 파일 이름을 넣어야 한다.

```
~
~
~
~
~
~
```

- 텍스트 입력 모드

  다음 줄로 이동하기 위해서 엔터 키를 누르면 된다.

  입력된 마지막 문자를 지우기 위해서 백스페이스 키를 이용할 수 있다.

  텍스트 입력 모드에서 명령 모드로의 전환은 Esc 키를 누름으로써 가능하다.

키	수행
i	텍스트가 커서 앞에서 삽입된다.
I	텍스트가 현재 줄의 시작에 삽입된다.
a	텍스트가 커서 뒤에서 삽입된다.
A	텍스트가 현재 줄의 끝에 삽입된다.
o	텍스트가 현재 줄 다음부터 삽입된다.
O	텍스트가 현재 줄 앞에서 삽입된다.
R	텍스트가 대치된다.(겹쳐 적힌다)

- 명령 모드

  vi의 편집 특성은 특수한 문자들을 연속적으로 누름으로써 선택된다.

범위	선택
1,$	화일 내의 모든 줄
1,.	화일의 첫 줄에서부터 현재 줄까지의 모든 줄
.,$	현재 줄에서부터 화일의 마지막 줄까지의 모든 줄
.-2	현재 줄부터 앞쪽으로 두번째에 해당하는 한 줄

실험 실습 결과보고서

구체적인 절차 기술
(Detail steps)

- 커서이동

이동	키 순서
한 줄 위로	<cursor up> 또는 k
한 줄 아래로	<cursor down> 또는 j
한 문자 오른쪽으로	<cursor right> 또는 l
한 문자 왼쪽으로	<cursor left> 또는 h
줄의 시작으로	^
줄의 마지막으로	$
한 단어 앞으로	b
한 단어 뒤로	w
반 화면 아래로	Control-D
한 화면 아래로	Control-F
반 화면 위로	Control-U
한 화면 위로	Control-B
줄 번호 nn 위치로	:nn<Enter>

- 텍스트의 삭제

지울 대상	키 순서
문자	문자 위에 커서를 위치시키고 "x"를 누른다.
단어	단어 시작 부분에 커서를 위치시키고 두 문자 "dw"를 누른다.
줄	줄 위의 아무 곳에나 커서를 위치시키고 두 문자 "dd"를 누른다. ("dd"앞에 수를 입력하면 현재 줄에서 시작하여 명시된 수 만큼 삭제된다.)
현재 줄의 끝까지	"D"를 누른다.
불특 단위의 줄	:<range>d<Enter>

- 텍스트의 치환

치환 대상	키 입력 순서
문자	문자 위에 커서를 위치시키고 "r"을 누른 다음 치환할 문자를 타이핑한다.
단어	단어의 첫 위치에 커서를 두고 "cw"를 누른 다음, 치환할 텍스트를 타이핑하고 Esc 키를 누른다.
줄	줄의 아무 위치에나 커서를 위치시키고 "cc"를 누른 다음, 치환할 텍스트를 타이핑하고 Esc 키를 누른다.

- 텍스트 붙이기

행동	키 순서
줄을 붙이기 버퍼로 복사(yank)	:<range>y<Enter>
현재 줄 다음에 붙이기 버퍼의 내용을 삽입(put)	p 또는 :pu<Enter> (붙이기 버퍼의 내용은 변하지 않는다)
줄 번호 nn번 다음에 붙이기 버퍼의 내용을 삽입(put)	:nnpu<Enter> (붙이기 버퍼의 내용은 변하지 않는다)

- 탐 색

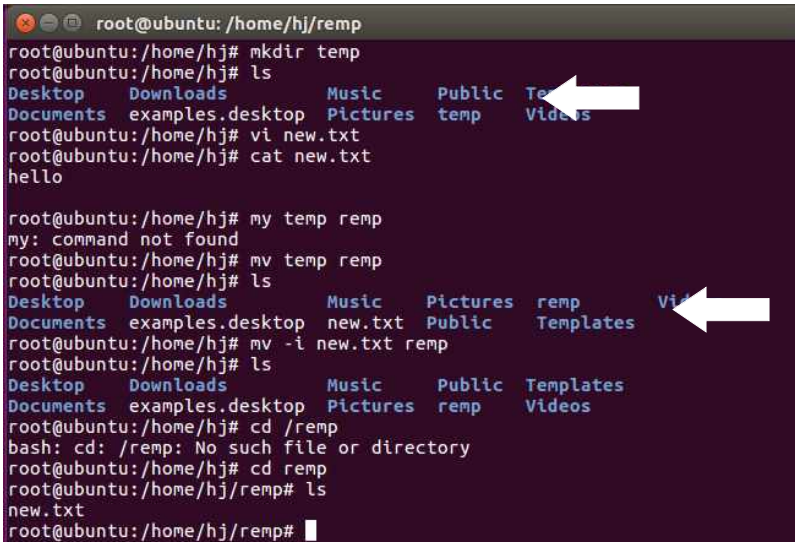
행동	키 순서
현재 위치에서 파일의 뒤쪽으로 문자열 sss를 탐색	/sss/<Enter>
현재 위치에서 파일의 앞쪽으로 문자열 sss를 탐색	?sss?<Enter>
마지막 탐색 명령을 반복	n
반대 방향으로 마지막 탐색 반복	N

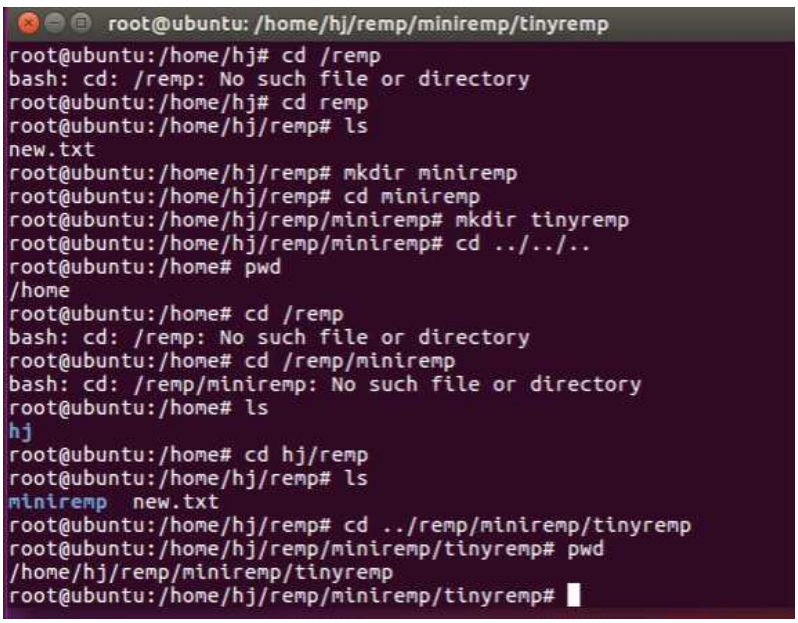
- 탐색치환

행동	키 순서
각 줄 중 sss가 처음 발견된 곳만 ttt로 치환시킨다.	:<range>s/sss/ttt/<Enter>
각 줄 중 sss가 발견되면 모두 ttt로 치환시킨다.	:<range>s/sss/ttt/g<Enter>

실험 결과
(Results)

mkdir –디렉토리 생성
ls – 디렉토리 내의 파일 확인
vi – vi 편집기. 존재하지않는 파일을 만들거나, 기존에 있던 파일로 들어간다. 중간중간 사용하였기에 따로 기술하지않았다.
vim-vi 편집기지만 vi보다 더 업그레이드 된 버전이다. vi보다 가독성이 높다.
cat – 파일의 내용을 출력한다.
mv – 파일의 이름변경/위치 이동
cd – 다른 디렉토리로 이동

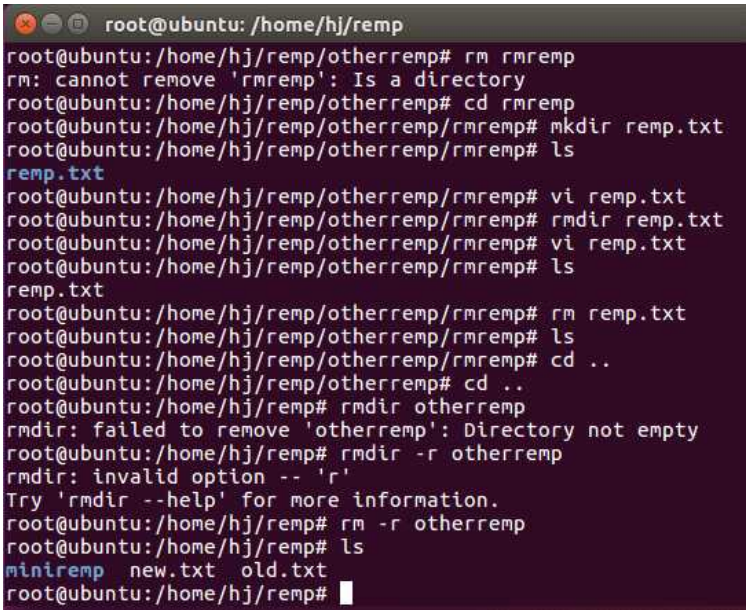




temp를 생성한다.(mkdir) 그리고 remp로 이름을 바꾸었다.(mv) home에 new.txt를 만들고(vi) 그것을 remp 파일로 옮겼다.(mv) remp/miniremp/tinyremp를 만들어서(mkdir) 디렉토리를 이동한다.(cd)

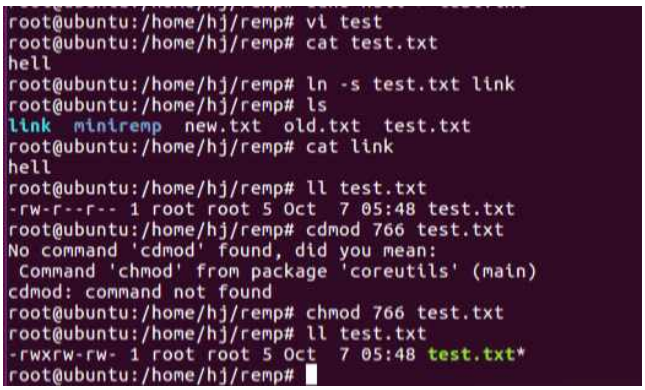
실험 결과
(Results)

rm- 파일을 지운다. 하위 디렉토리까지 삭제하고 싶으면 적절한 옵션을 넣는다.
rmdir -디렉토리를 삭제한다.



단순히 rm rmremp를 하면 rmremp가 디렉토리기 때문에 삭제가 되지않는다. 하위 디렉토리가 없는 디렉토리를 삭제할때 rmdir을 사용한다. 하위 디렉토리까지 삭제하고싶으면 옵션 -r을 붙여 rm -r을 하여 전부 삭제한다. otherremp 안의 텍스트파일까지 전부 삭제된다.

link -파일과 파일간의 링크를 만든다.
chmod - 파일의 허가권을 바꾼다.



test.txt의 링크를 link라는 이름으로 만들었다. 그래서 link를 읽으면 test와 같은 내용이 나온다.
test.txt의 초기 허가권은 rw-r- -r- -으로, owner의 권한은 read write, group user와 etc user의 권한은 read만이었다. chmod를 사용하여 rwxrw-rw-를 만들었다.

실험 결과
(Results)

ping – 네트워크 관리 명령어. ip를 확인할 수 있다.
ifconfig –네트워크 인터페이스를 설정하거나 확인할 수 있다.

```
root@ubuntu:/home/hj/remp# ping www.naver.com
PING e6030.a.akamaiedge.net (173.223.141.49) 56(84) bytes of data.
64 bytes from a173-223-141-49.deploy.static.akamaitechnologies.com (173.223.141.
49): icmp_seq=1 ttl=128 time=135 ms
64 bytes from a173-223-141-49.deploy.static.akamaitechnologies.com (173.223.141.
49): icmp_seq=2 ttl=128 time=134 ms
^C
--- e6030.a.akamaiedge.net ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 134.267/134.752/135.238/0.608 ms
root@ubuntu:/home/hj/remp#
```

```
root@ubuntu:/home/hj/remp# ifconfig
ens33     Link encap:Ethernet  HWaddr 00:0c:29:24:8c:aa
          inet addr:192.168.145.128  Bcast:192.168.145.255  Mask:255.255.255.0
          inet6 addr: fe80::8077:f183:9f49:6778/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:81 errors:0 dropped:0 overruns:0 frame:0
          TX packets:167 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
```

www.naver.com의 ping을 확인하였다.
ifconfig를 통해 Mask : 255.255.255.0을 확인 했다.

gzip – 파일을 gz.으로 압축한다. 하위 파일이 있으면 압축이 불가하다.
tar– 파일 혹은 디렉토리를 원하는 확장자로 압축하거나 푼다. gzip보다 더 유용하여 주로 tar을 쓴다.

```
root@ubuntu:/home/hj/remp/miniremp# gzip new2.txt
root@ubuntu:/home/hj/remp/miniremp# ls
new2.txt.gz  tinyremp
root@ubuntu:/home/hj/remp/miniremp#
```

```
root@ubuntu:/home/hj/remp/miniremp# tar zcvf tinyremp.tar tinyremp
tinyremp/
tinyremp/newtxt
tinyremp/new.txt
root@ubuntu:/home/hj/remp/miniremp# ls
new2.txt.gz  tinyremp  tinyremp.tar
root@ubuntu:/home/hj/remp/miniremp# tar zxvf tinyremp.tar
tinyremp/
tinyremp/newtxt
tinyremp/new.txt
root@ubuntu:/home/hj/remp/miniremp# ls
new2.txt.gz  tinyremp  tinyremp.tar
root@ubuntu:/home/hj/remp/miniremp# tar zxvf tinyremp.tar -C ./remp
tar: remp: Cannot open: No such file or directory
tar: Error is not recoverable: exiting now
root@ubuntu:/home/hj/remp/miniremp# ^C
root@ubuntu:/home/hj/remp/miniremp# tar zxvf tinyremp.tar -C ./remp
tar: remp: Cannot open: No such file or directory
tar: Error is not recoverable: exiting now
root@ubuntu:/home/hj/remp/miniremp# tar zxvf tinyremp.tar -C ./remp/
tar: remp: Cannot open: No such file or directory
tar: Error is not recoverable: exiting now
root@ubuntu:/home/hj/remp/miniremp# rm -r tinyremp
root@ubuntu:/home/hj/remp/miniremp# ls
new2.txt.gz  tinyremp.tar
root@ubuntu:/home/hj/remp/miniremp# tar zxvf tinyremp.tar
tinyremp/
tinyremp/newtxt
tinyremp/new.txt
root@ubuntu:/home/hj/remp/miniremp# ls
new2.txt.gz  tinyremp  tinyremp.tar
```

tinyremp를 tar zcvf로 하위 파일과 함께 전부 tinyremp.tar로 압축하였다. 그리고 zxvf로 tinyremp.tar을 압축해제 하여 원래의 파일을 만들었다.

실험 결과
(Results)

df- 용량을 체크해주는 명령어
adduser - 리눅스에 사용자 추가
passwd-사용자 계정에 패스워드를 만들거나 변경한다.

```
root@ubuntu:/home/hj/remp/miniremp# adduser hj2
Adding user `hj2' ...
Adding new group `hj2' (1001) ...
Adding new user `hj2' (1001) with group `hj2' ...
Creating home directory `/home/hj2' ...
Copying files from `/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for hj2
Enter the new value, or press ENTER for the default
        Full Name []: hjl
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [Y/n] y
root@ubuntu:/home/hj/remp/miniremp# passwd hj2
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
```

```
root@ubuntu:/home/hj/remp/miniremp# df
Filesystem     1K-blocks    Used Available Use% Mounted on
udev              996032       0    996032   0% /dev
tmpfs             203060    6436    196624   4% /run
/dev/sda1       18447100 3897040  13589960  23% /
tmpfs            1015300     288   1015012   1% /dev/shm
tmpfs               5120       4      5116   1% /run/lock
tmpfs            1015300       0   1015300   0% /sys/fs/cgroup
tmpfs             203060      56    203004   1% /run/user/1000
```

du - 리눅스 시스템 내에있는 디렉토리와 파일개수를 출력한다.
echo - 문장 또는 시스템 환경변수를 바로 출력한다.
free - 시스템의 메모리, 사용된 물리적인 메모리와 스왑 메모리의 상태를 출력한다.

```
root@ubuntu:/home/hj/remp/miniremp# du
4       ./tinyremp
16      .
root@ubuntu:/home/hj/remp/miniremp# echo maaaaan
maaaaan
root@ubuntu:/home/hj/remp/miniremp# free
              total        used        free      shared  buff/cache   available
Mem:        2030600      716516      743272       13180      570812     1109080
Swap:       2094076           0     2094076
```

실험 결과
(Results)

grep – 파일이나 디렉토리에서 원하는 글자를 찾게해주는 명령어.

```
root@ubuntu:/home/hj/remp# cat rrr

im hyunjung
i like game and cartoon


first,
i dont know what i do
hate study :(




root@ubuntu:/home/hj/remp# grep game ./rrr
i like game and cartoon
root@ubuntu:/home/hj/remp# vi new.txt
root@ubuntu:/home/hj/remp# vi rrr
root@ubuntu:/home/hj/remp# grep -r hello ./remp
grep: ./remp: No such file or directory
root@ubuntu:/home/hj/remp# grep -r hello
new.txt:hello
rrr:hello
old.txt:hello
root@ubuntu:/home/hj/remp# cd ..
root@ubuntu:/home/hj# grep -r hello ./remp
./remp/new.txt:hello
./remp/rrr:hello
./remp/old.txt:hello
root@ubuntu:/home/hj#
```

rrr이라는 파일안의 game이라는 글자를 찾게 했다.
또한 grep –r을 이용하여, remp라는 디렉토리 내의 hello라는 글자를 하위 디렉토리 내에서 전부 찾을 수 있었다.

history – 기존에 실행하였던 명령어들을 보여준다.
ps – 현재 구동되고있는 프로세스들을 보여준다.

```
root@ubuntu:/home/hj# history 10
  139  cat rrr
  140  grep game ./rrr
  141  vi new.txt
  142  vi rrr
  143  grep -r hello ./remp
  144  grep -r hello
  145  cd ..
  146  grep -r hello ./remp
  147  history
  148  history 10
root@ubuntu:/home/hj# ps -9
    PID TTY      STAT   TIME COMMAND
root@ubuntu:/home/hj# ps
    PID TTY          TIME CMD
   5026 pts/1    00:00:00 su
   5028 pts/1    00:00:00 bash
   6046 pts/1    00:00:00 ps
```

실험 결과
(Results)

kill – 현재 구동되고있는 프로세서들을 종료한다.
pwd – 현재 디렉토리를 확인한다.

```
root@ubuntu:/home/hj# ps
   PID TTY          TIME CMD
  6109 pts/1    00:00:00 su
  6110 pts/1    00:00:00 bash
  6120 pts/1    00:00:00 ps
root@ubuntu:/home/hj# kill -9 ps
bash: kill: ps: arguments must be process or job IDs
root@ubuntu:/home/hj# kill -9 6120
bash: kill: (6120) - No such process
root@ubuntu:/home/hj# kill 6120
bash: kill: (6120) - No such process
root@ubuntu:/home/hj# ps
   PID TTY          TIME CMD
  6109 pts/1    00:00:00 su
  6110 pts/1    00:00:00 bash
  6123 pts/1    00:00:00 ps
root@ubuntu:/home/hj# kill -9 6110
hj@ubuntu:~$ su root
Password:
root@ubuntu:/home/hj# pwd
/home/hj
```

:wq , :q! – vi 종료 명령어. wq는 저장하고 정리하고 q!는 강제 종료이다.
:set num – vi 명령어로, vi 코드에 번호를 붙여준다.

wq를 하고 나갔기 때문에 helloworld를 그대로 출력하였다. 이 코드에 :set nu를 하였더니 줄마다 번호가 매겨졌다.
그 뒤에 !q test를 추가한 코드에 q!를 하고 나갔더니 저장되지않아 그 전 코드가 그대로 나왔다.

실험 후기
(느낀 점)

리눅스가 많은 명령어를 가지고있고, 사용자가 자유롭게 설정을 바꿀수있다는 점에서 굉장히 흥미롭고 재미있지만, 아직은 이것을 어떻게 임베디드 소프트웨어를 제어할수있을지 잘 감이 오지않는다. 또한 명령어가 무슨 기능을 하는지는 알지만 특정한 프로그램을 짜보진않아서 어떻게 응용을 할수있을지 기대가 된다. 앞으로의 수업에서 임베디드 장치를 연결하여 실습해보면서 이것을 배우고 익히고싶다는 생각을 하였다.