



1장 리눅스의 기본 개념과 프로그램 작성

- 리눅스의 기본 개념
- 시스템에 접속하기
- 리눅스의 프로그래밍 환경
 - vi : 텍스트 에디터
 - gcc : C Compiler
 - make, Makefile
 - gdb : 디버거

● 다중 작업 (multi process)

- 선점 가능한(preemptive) 실제 다중 작업을 지원
- 작업은 실행 중인 상태의 프로그램을 의미 (프로세스)

● 다중 사용자 (multi user)

- 동시에 여러 명의 사용자가 시스템에서 작업하는 것을 허용
- 터미널이나 네트워크 연결을 통해서 동일한 하나의 리눅스 시스템을 사용

● 다중 프로세서 (multi processor)

- 다중 프로세서 구조에서도 실행될 수 있음 (두 개 이상의 CPU를 가진 시스템을 리눅스가 지원)

● 이식성과 확장성

- 이식성이 높음
- 다양한 언어로 작성된 프로그램을 어렵지 않게 사용할 수 있음

● 파일 시스템

- 리눅스의 파일 시스템은 유닉스의 것과 같이 트리 구조를 이루고 있음
- 별도로 추가된 물리적인 보조 기억 장치들이나 하드웨어 디바이스들도 파일 형태로 파일 시스템에 연결

● 권한

- 사용자별로 별도의 권한을 부여
- 하나의 시스템을 여러 명의 사용자가 동시에 사용할 수 있기 때문에 발생할 수 있는 여러 가지 문제를 사전에 방지
- 시스템을 관리하기 위한 관리자와 시스템을 사용하기만 하는 사용자

● 셸

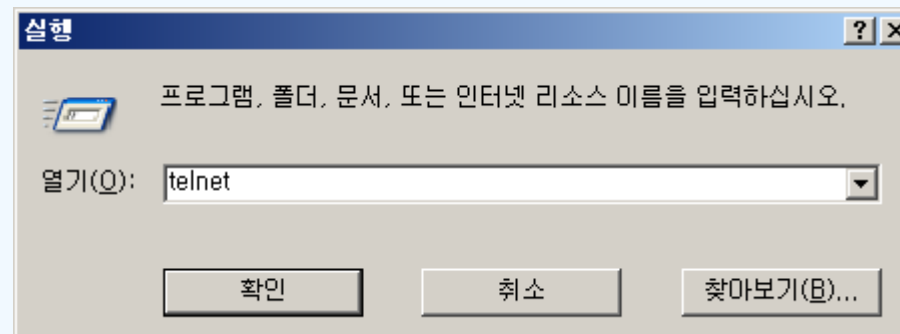
- 사용자가 시스템을 쉽게 사용할 수 있도록 중간자 역할의 프로그램
- 사용자가 명령어 라인을 입력해서 원하는 작업을 수행할 수 있음

● 개발 환경

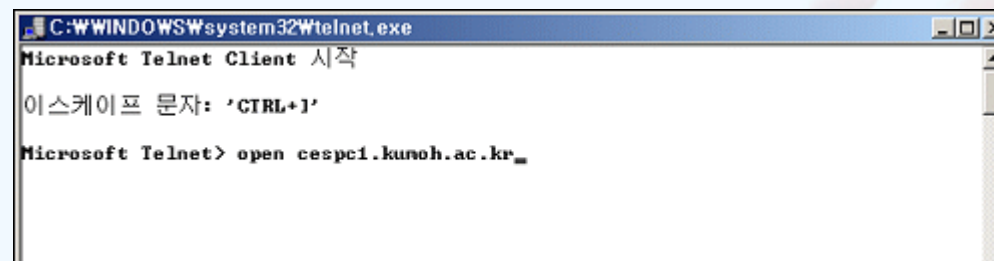
- 리눅스는 새로운 프로그램을 개발하기 위한 환경을 제공
- 프로그래밍 언어용 컴파일러, 프로그램 개발에 필요한 보조적인 유틸리티

● 리눅스 시스템의 콘솔이나 터미널과 같은 원격 장치를 사용

- 시스템의 콘솔은 시스템 관리자만 선택할 수 있는 방법
- 일반 사용자는 터미널을 사용하거나 텔넷(Telnet)과 같은 응용 프로그램을 사용

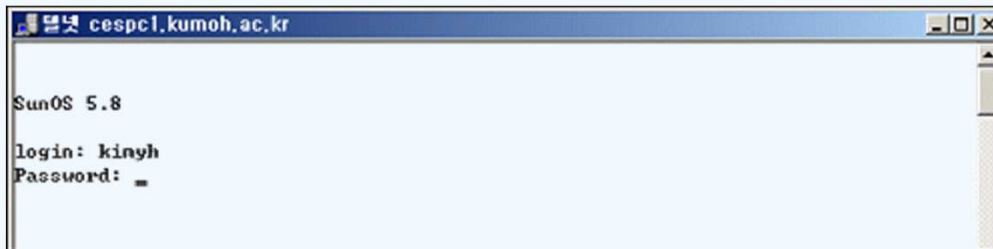


telnet 실행 후 open 명령어 사용



● 계정 정보 입력

- ➔ 유닉스나 리눅스와 같이 다중 사용자를 지원하는 시스템을 사용하기 위해서는 해당 시스템에 사용자로 등록되어 있어야 함
- ➔ 계정은 ID와 암호를 의미



```
네트워크 cespcl.kumoh.ac.kr

SunOS 5.8

login: kinyh
Password: _
```

로그인에 성공하면 셸 프롬프트가 나타난다.



```
네트워크 cespcl.kumoh.ac.kr

SunOS 5.8

login: kinyh
Password:
Last login: Mon Jan  8 11:06:58 from 202.31.201.117
Sun Microsystems Inc. SunOS 5.8 Generic Patch October 2001
[unknown:kinyh 1 ] _
```

왼쪽의 그림은
유닉스 시스템
에 접속한 예
다.

● 리눅스 시스템에 접속

리눅스 시스템은 보안 기능이 지원되는 터미널 프로그램으로 접속하는 것이 일반적임

```
ce.kumoh.ac.kr - PuTTY
로그인: kimyh
kimyh@ce.kumoh.ac.kr 의 비밀번호:
Last login: Mon Jan  8 11:16:56 2007 from 202.31.201.117
-bash-3.00$
```

PuTTY를 사용
하여 리눅스
시스템에 접속
하는 예

```
ce.kumoh.ac.kr - PuTTY
로그인: kimyh
kimyh@ce.kumoh.ac.kr 의 비밀번호:
Last login: Mon Jan  8 11:16:56 2007 from 202.31.201.117
-bash-3.00$ ls
Desktop book code lecture script temp
-bash-3.00$ ls -l
한계 24
drwxr-xr-x  3 kimyh graduated 4096 1월  8 03:29 Desktop
drwxr-xr-x 14 kimyh graduated 4096 1월  8 11:16 book
drwxr-xr-x  2 kimyh graduated 4096 1월  8 11:16 code
drwxr-xr-x  5 kimyh graduated 4096 1월  8 11:16 lecture
drwxr-xr-x  2 kimyh graduated 4096 1월  8 11:16 script
-rw-r--r--  1 kimyh graduated   5 1월  8 11:16 temp
-bash-3.00$
```

로그인 후 ls
명령을 사용한
예

※MS Windows의 텔넷(telnet)과 PuTTY는 유닉스나 리눅스에 접속할 수 있는 터미널이다. 접속하려는 시스템의 보안 정책에 따라 적절한 것을 선택한다.

● 리눅스의 명령어 사용하기

리눅스 시스템의 명령을 사용하려면 해당 명령의 이름을 셸 프롬프트 상에서 입력한다.

```
ce.kumoh.ac.kr - PuTTY
-bash-3.00$ id
uid=504(kimyh) gid=501(graduated) groups=501(graduated)
-bash-3.00$ id kimyh
uid=504(kimyh) gid=501(graduated) groups=501(graduated)
-bash-3.00$
```

사용자의 ID와
그룹을 확인하
는 “id” 명령어
의 사용 예

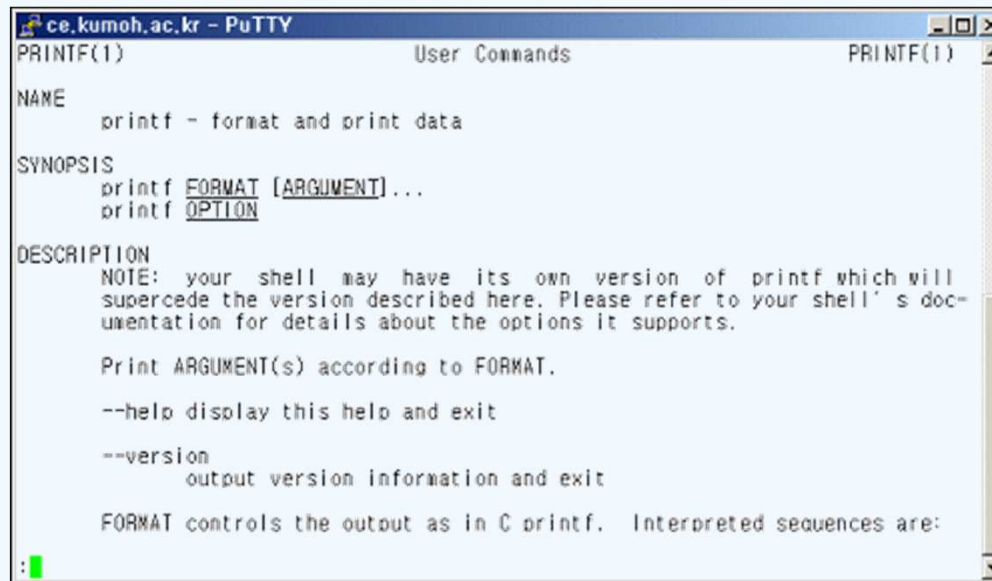
```
ce.kumoh.ac.kr - PuTTY
ID(1)                                User Commands                                ID(1)
NAME
  id - print real and effective UIDs and GIDs
SYNOPSIS
  id [OPTION]... [USERNAME]
DESCRIPTION
  Print information for USERNAME, or the current user.
  -a      ignore, for compatibility with other versions
  -Z, --context
          print only the context
  -g, --group
          print only the effective group ID
  -G, --groups
          print all group IDs
  -n, --name
```

“man id”를 실행하여 id 명령
의 사용법을
확인하는 예

● man 명령어의 사용 예

- ➔ man 명령어는 리눅스 시스템의 명령어 뿐만 아니라 프로그래밍 언어의 함수나 셸 스크립트의 사용법도 확인할 수 있다.

- ➔ “man printf”를 실행하여 printf의 사용법을 확인하기



```
ce.kumoh.ac.kr - PuTTY
PRINTF(1)                                User Commands  PRINTF(1)

NAME
    printf - format and print data

SYNOPSIS
    printf FORMAT [ARGUMENT]...
    printf OPTION

DESCRIPTION
    NOTE: your shell may have its own version of printf which will
    supercede the version described here. Please refer to your shell's doc-
    umentation for details about the options it supports.

    Print ARGUMENT(s) according to FORMAT.

    --help display this help and exit

    --version
        output version information and exit

    FORMAT controls the output as in C printf. Interpreted sequences are:
```

man 명령은 시스템의 명령어나 프로그래밍 언어의 함수 사용법을 익히기 위한 핵심이다.

● 프로그래밍 관련 툴

➔ 에디터 (editor)

- ▶ 소스 코드를 편집하는 용도로 사용된다.
- ▶ 일반적으로 vi 에디터를 사용한다.

➔ 컴파일러 (compiler)

- ▶ 소스 코드를 바이너리 코드로 변경한다.
- ▶ 리눅스 시스템에서는 gcc를 사용한다.

➔ 링커/로더 (linker/loader)

- ▶ 목적(object) 파일들을 연결해서 실행 파일을 만든다.
- ▶ 리눅스 시스템에서는 ld를 사용한다.
- ▶ 대부분 컴파일러가 알아서 실행해준다.



● 소스 코드 편집하기

➔ 소스 코드를 편집하기 위한 두 가지 방법

- ▶ PC에서 편집하여 FTP 서비스로 리눅스 시스템에 업로드
- ▶ 리눅스 시스템에서 직접 편집

➔ 리눅스 시스템에서 직접 편집

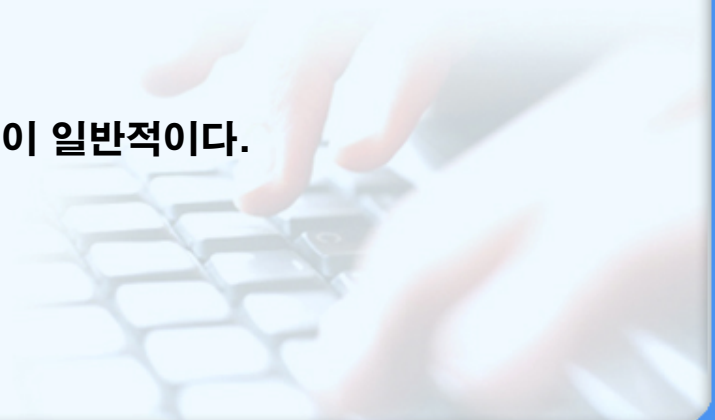
- ▶ 간단한 소스 코드

```
$ cat > hello.c
```

- ▶ 복잡한 소스 코드

```
$ vi hello.c
```

- ▶ ※vi 편집기를 사용하여 소스 코드를 편집하는 것이 일반적이다.



● 컴파일하기

➔ gcc (GNU C Compiler)

- ▶ 무료 배포가 원칙인 리눅스에서 일반적으로 사용하는 공개판 C 컴파일러
- ▶ 리눅스를 설치할 때 같이 설치할 수 있다.

➔ 설치 확인

- ▶ 셸 프롬프트 상태에서 gcc를 실행해본다.



```
ce.kumoh.ac.kr - PuTTY
-bash-3.00$ vi hello.c
-bash-3.00$ ls
hello.c
-bash-3.00$ gcc
gcc: no input files
-bash-3.00$
```

gcc: no input files 라는 메시지가 출력되면 설치되어 있다.

● gcc 사용하기

➔ `$ gcc [options] source_files`

▶ options

`-o output_filename` : 실행파일을 만들 경우 실행 파일의 이름을 지정한다.

`-c` : 지정한 소스코드의 목적(object) 파일을 만든다.

▶ source_files

.c 를 확장자로 가지는 소스 코드(들)

▶ 사용 예

`$ gcc hello.c`

`$ gcc -o hello hello.c`

\$ man gcc 를 실행하여
gcc의 자세한 사용법을
확인할 수 있다.

● gcc 사용 예

hello.c를 컴파일하여 실행 파일을 만든다.

ls 명령으로 생성된 파일을 확인한다.

```
$ gcc hello.c
$ ls -l
-rwxr-xr-x  1 kimyh  graduate 13508 Nov 18 15:05 a.out
-rw-r--r--  1 kimyh  graduate  58 Nov 18 15:04 hello.c
$ ./a.out
hello world!
$
```

gcc를 실행할 때 출력 파일의 이름을 지정하지 않았기 때문에 실행 파일의 이름이 a.out이다.

a.out을 실행하고 결과를 확인한다.

● 컴파일 시 오류 발생 및 해결

- ➔ 소스 코드를 컴파일할 때 오류가 있을 경우 gcc는 오류 메시지를 출력한다.
- ➔ 오류가 발생하면 소스 코드를 수정하고 다시 컴파일한다.

```
$ cat hello.c
#include <stdio.h>
```

```
main()
{
    printf("hello world!\n")
}
```

문장이 ';'으로 종결되지 않았다.

```
$ gcc -o hello hello.c
hello.c: In function 'main':
hello.c:6: parse error before `}'
$
```

컴파일을 한 결과 소스 코드의 6번 라인에서 오류가 생겼다.

● 두 개 이상의 소스 코드로 하나의 실행 파일 만들기

큰 규모의 프로그램은 여러 개의 소스 코드로 나누어서 작성하는 것이 일반적이다.

one.c	two.c
<pre>#include <stdio.h> void printmsg(void); main() { printmsg(); }</pre>	<pre>#include <stdio.h> void printmsg(void) { printf("hello world!\n"); }</pre>

\$ gcc -o three one.c two.c

출력 파일의 이름은 three이다.

소스 코드 파일은 2개로 각각 one.c와 two.c다.

● make와 Makefile

➔ make

- ▶ Makefile에서 정한 파일 연관성과 생성 순서에 따라 컴파일(compile)과 링킹(linking) 작업을 수행하여 실행 파일을 작성한다.

➔ Makefile

- ▶ 하나의 실행 파일을 작성하기 위해서 관련된 소스 코드 파일들을 컴파일하고 링크하는 순서를 정의하고 있는 텍스트 형식의 파일이다.

▶ 형식

```
target_list: dependency_list  
            command_list
```

▶ 예제

```
three: one.c two.c  
      gcc -o three one.c two.c
```

● Makefile의 예

생성할 파일의 이름

three를 생성하기 위해 필요한
소스 코드 파일들

```
three: one.c two.c
gcc -o three one.c two.c
```

명령어 라인은 반드시
Tab으로 들여쓰기 한다.

one.c와 two.c를 사용하여 three를
생성하기 위해 실행해야 하는
명령어 라인

● Makefile의 이름

기본적으로 Makefile이란 이름을 사용하나 다른 이름을 사용할 수 있다.

```
$ make -n makefile_hello
```

-n 옵션으로 지정할 수 있다.

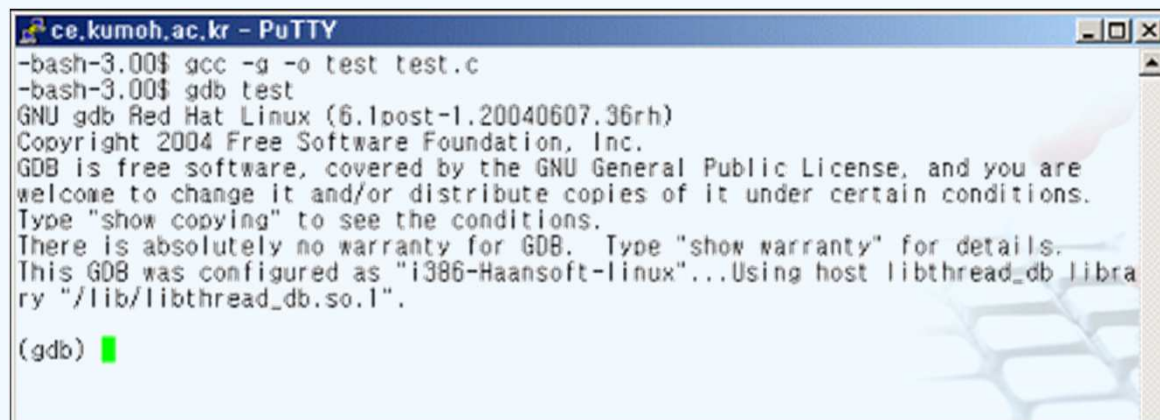
● 디버깅하기 (1)

➔ GNU gdb

- ▶ 소스 코드의 명령어들을 하나씩 처리하면서 프로그램이 실행되는 과정을 확인할 수 있는 도구이다.
- ▶ gdb를 사용하기 위해서는 gcc를 반드시 -g 옵션을 사용하여 실행파일을 만들어야 한다.

```
$ gcc -g -o test test.c
```

➔ gdb 실행 예



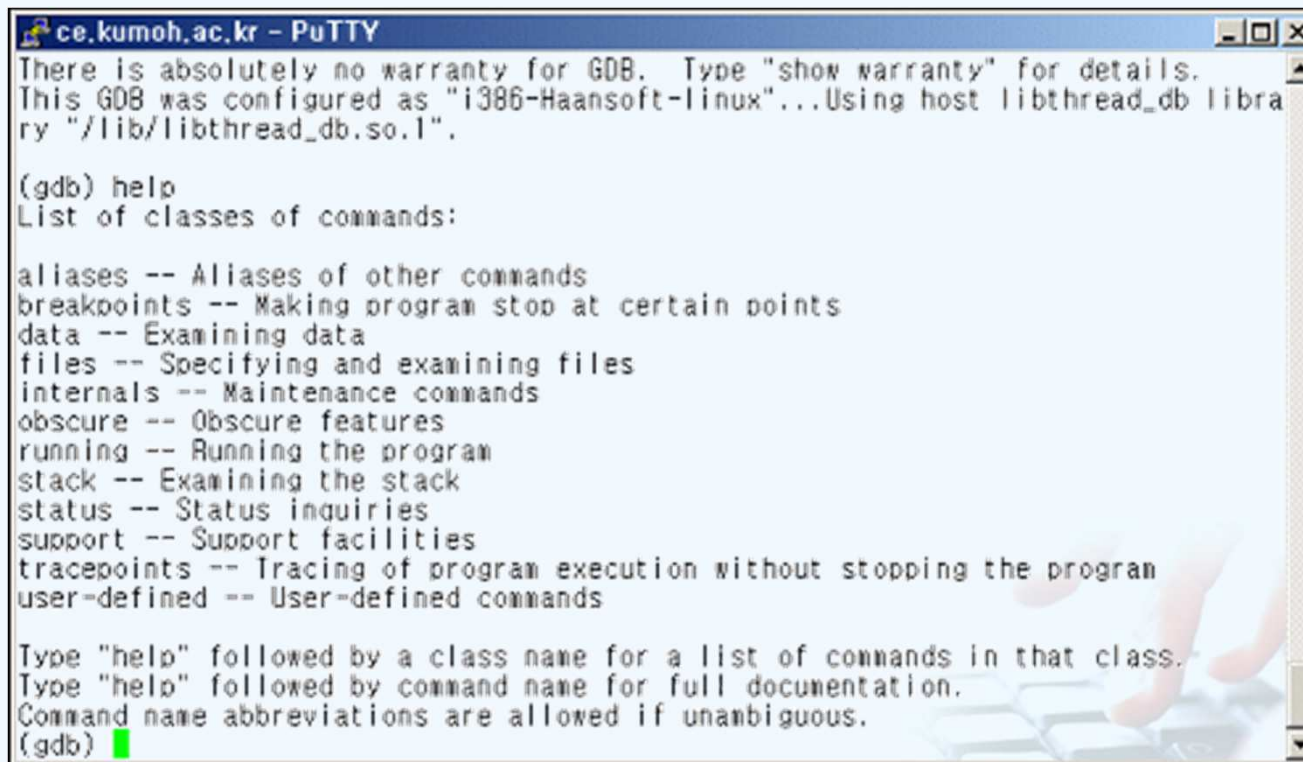
```
ce.kumoh.ac.kr - PuTTY
-bash-3.00$ gcc -g -o test test.c
-bash-3.00$ gdb test
GNU gdb Red Hat Linux (6.1post-1.20040607.36rh)
Copyright 2004 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public License, and you are
welcome to change it and/or distribute copies of it under certain conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB. Type "show warranty" for details.
This GDB was configured as "i386-Haansoft-linux"...Using host libthread_db libra
ry "/lib/libthread_db.so.1".

(gdb) █
```

● 디버깅하기 (2)

gdb의 명령어 확인

- ▶ gdb를 실행 중인 상태에서 help를 입력한다.



```
ce.kumoh.ac.kr - PuTTY
There is absolutely no warranty for GDB.  Type "show warranty" for details.
This GDB was configured as "i386-Haansoft-linux"...Using host libthread_db library "/lib/libthread_db.so.1".

(gdb) help
List of classes of commands:

aliases -- Aliases of other commands
breakpoints -- Making program stop at certain points
data -- Examining data
files -- Specifying and examining files
internals -- Maintenance commands
obscure -- Obscure features
running -- Running the program
stack -- Examining the stack
status -- Status inquiries
support -- Support facilities
tracepoints -- Tracing of program execution without stopping the program
user-defined -- User-defined commands

Type "help" followed by a class name for a list of commands in that class.
Type "help" followed by command name for full documentation.
Command name abbreviations are allowed if unambiguous.
(gdb) █
```