오픈소스소프트웨어

Open-Source Software

ICT융합학부 조용우

파일 생성 및 편집

생성 파일 출력 : touch

$ touch 파일

파일 크기가 0인 이름만 있는 빈 파일을 만들어 준다.

```
$ touch cs2.txt
$ ls -asl cs2.txt
0 -rw-rw-r--. 1 ywcho ywcho 0 3월 26 15:10 cs2.txt.
```

페이지 단위로 파일 내용 보기 : more

$ more 파일

파일(들)의 내용을 페이지 단위로 화면에 출력한다.

```
$ more cs3.txt
Unix is a multitasking, multi-user computer operating system originally
developed in 1969 by a group of AT&T employees at Bell Labs, including
Ken Thompson, Dennis Ritchie, Brian Kernighan, Douglas McIlroy,
and Joe Ossanna.
...
During the late 1970s and early 1980s, the influence of Unix in academic
circles led to large-scale adoption of Unix(particularly of the BSD variant,
--계속--(59%)
```

파일 앞부분 보기 : head

$ head [-n] 파일

파일(들)의 앞부분을 화면에 출력한다. 파일을 지정하지 않으면 표준입력 내용을 대상으로 한다.

```
$ head -5 cs3.txt
Unix is a multitasking, multi-user computer operating system originally
developed in 1969 by a group of AT&T employees at Bell Labs, including
Ken Thompson, Dennis Ritchie, Brian Kernighan, Douglas McIlroy,
and Joe Ossanna.
```

파일 뒷부분 보기 : tail

$ tail [-n] 파일

파일(들)의 뒷부분을 화면에 출력한다. 파일을 지정하지 않으면 표준입력 내용을 대상으로 한다.

```
$ tail cs3.txt
Linux, which is used to power data centers, desktops, mobile phones,
and embedded devices such as routers, set-top boxes or e-book readers.
Today, in addition to certified Unix systems such as those already
mentioned, Unix-like operating systems such as MINIX, Linux, Android,
and BSD descendants (FreeBSD, NetBSD, OpenBSD, and DragonFly BSD) are
commonly encountered.
The term traditional Unix may be used to describe a Unix or
an operating system that has the characteristi cs of either Version 7
Unix or UNIX System V.
```

파일 사용

파일 복사 : cp(copy)

$ cp [-i] 파일1 파일2

파일 1을 파일 2에 복사한다. –i는 대화형 옵션이다

```
$ cp cs3.txt cs2.txt
$ ls -l cs3.txt cs2.txt
-rw-r--r-- 1 ywcho cs 2088 3월 16일 13:37 cs3.txt
-rw-r--r-- 1 ywcho cs 2088 3월 16일 13:45 cs2.txt
$ cp /etc/hosts hostnames
```

파일 복사 : cp(copy)

$ cp [-i] 파일1 파일2

파일 1을 파일 2에 복사한다. –i는 대화형 옵션이다

- 대화형 옵션: cp -i
  - 복사 대상 파일과 이름이 같은 파일이 이미 존재하면 덮어쓰기(overwrite)
  - 보다 안전한 사용법: 대화형 –i(interactive) 옵션을 사용

```
$ cp –i cs1.txt cs2.txt
cp: overwrite ‘cs2.txt’? n
```

파일 복사 : cp(copy)

$ cp 파일 디렉터리

파일을 지정된 디렉터리에 복사한다

$ cp 파일1 … 파일 n 디렉터리

여러 개의 파일들을 지정된 디렉터리에 모두 복사한다

```
$ cp cs1.txt /tmp
$ ls -l /tmp/cs1.txt
-rw-r--r-- 1 ywcho cs 2088 4월 16일 14:31 /tmp/cs1.txt
$ cp cs1.txt cs2.txt /tmp
```

파일 복사 : cp(copy)

$ cp [-r] 디렉터리1 디렉터리2

R은 리커전 옵션으로 디렉터리 1 전체를 디렉터리 2에 복사한다

- 하위 디렉터리를 포함한 디렉터리 전체를 복사

```
$ cp -r test temp
```

파일 이동 : mv(move)

$ mv [-i] 파일1 파일2

파일 1의 이름을 파일 2로 변경한다. –i는 대화형 옵션이다

```
$ mv cs2.txt cs4.txt
$ ls -l
-rw-r--r-- 1 ywcho cs 2088 3월 16일 13:37 cs1.txt
-rw-r--r-- 1 ywcho cs 2088 3월 16일 13:56 cs4.txt
-rw-r--r-- 1 ywcho cs 2088 3월 16일 13:56 cs3.txt
```

파일 이동 : mv(move)

$ mv [-i] 파일1 파일2

파일 1의 이름을 파일 2로 변경한다. –i는 대화형 옵션이다

- 대화형 옵션: cp -i
  - 이동 대상 파일과 이름이 같은 파일이 이미 존재하면 덮어쓰기(overwrite)
  - 보다 안전한 사용법: 대화형 –i(interactive) 옵션을 사용

```
$ mv –i cs1.txt cs3.txt
mv: overwrite ‘cs3.txt’? n
```

파일 이동 : mv(move)

$ mv 파일 디렉터리

파일을 지정된 디렉터리에 복사한다

$ cp 파일1 … 파일 n 디렉터리

여러 개의 파일들을 지정된 디렉터리에 모두 복사한다

```
$ mv cs3.txt /tmp
$ ls -l /tmp/cs3.txt
-rw-r--r-- 1 ywcho cs 2088 3월 16일 13:56 /tmp/cs3.txt
$ mv cs1.txt cs3.txt /tmp
```

파일 이동 : mv(move)

$ mv 디렉터리1 디렉터리2

디렉터리 1의 지정된 디렉터리 2로 이름을 변경한다.

```
$ mkdir temp
$ mv temp tmp
```

파일 삭제 : rm(remove)

$ rm [-i] 파일

파일(들)을 삭제한다. –i는 대화형 옵션이다.

```
$ rm cs1.txt
$ rm cs2.txt cs3.txt
```

- 대화형 옵션 : rm -i

```
$ rm –i cs1.txt
rm: remove 'cs1.txt‘ ? n
```

파일 삭제 : rm(remove)

$ rm [-ri] 디렉터리

-r은 리커젼 옵션으로 디렉터리 아래의 모든 것을 삭제한다.

-i는 대화형 옵션

```
$ rm test
rm: cannot remove 'test': 디렉터리입니다
$ rmdir test
rmdir: failed to remove 'test': 디렉터리가 비어있지 않음
$ rm -ri test
rm: descend into directory 'test'? y
rm: remove regular file 'test/cs3.txt'? y
rm: remove regular file ‘test/cs1.txt’? y
rm: remove directory 'test'? y
```

링크(Link)

링크 : ln(link)

$ ln [-s] 파일1 파일2

파일 1에 대한 새로운 이름(링크)로 파일 2를 만들어 준다.

-s 옵션은 심볼릭 링크

$ ln [-s] 파일1 디렉터리

파일 1에 대한 링크를 지정된 디렉터리에 같은 이름으로 만들어 준다.

심볼릭 링크(symbolic link)

- 심볼릭 링크
  - 다른 파일을 가리키고 있는 별도의 파일이다.
  - 실제 파일의 경로명을 저장하고 있는 일종의 특수 파일이다.
  - 이 경로명이 다른 파일에 대한 간접적인 포인터 역할을 한다.

```
$ ln -s hello.txt hi.txt
$ ls -l
-rw------- 1 ywcho cs 15 3월 16일 15:31 hello.txt
lrwxrwxrwx 1 ywcho cs 9 1월 24일 12:56 hi.txt -> hello.txt

$ ln -s /usr/bin/gcc cc
$ ls -l cc
lrwxrwxrwx. 1 chang chang 12 7월 21 20:09 cc -> /usr/bin/gcc
```

하드 링크(hard link)

- 하드 링크
  - 기존 파일에 대한 새로운 이름이라고 생각할 수 있다.
  - 실제로 기존 파일을 대표하는 i-노드를 가리켜 구현한다.

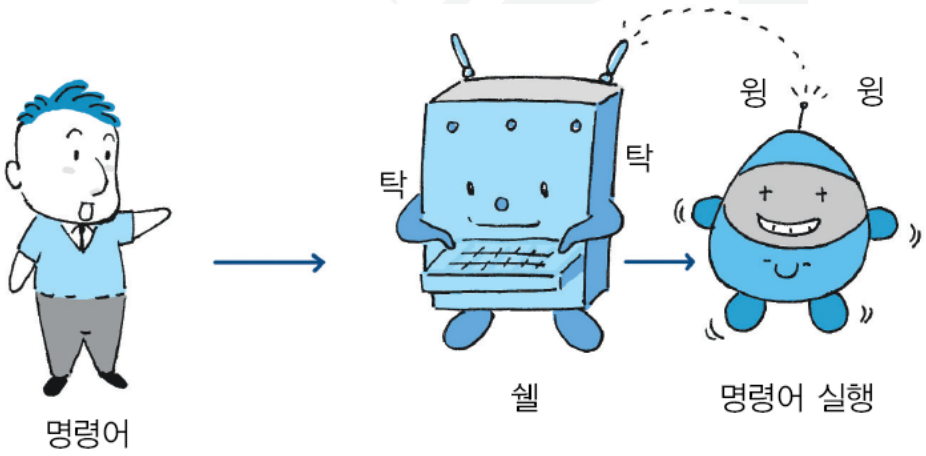
```
$ ln hello.txt hi.txt
$ ls -l
-rw------- 2 chang cs 15 11월 7일 15:31 hello.txt
-rw------- 2 chang cs 15 11월 7일 15:31 hi.txt
```

- 질문
  - 이 중에 한 파일의 내용을 수정하면 어떻게 될까?
  - 이 둘 중에 한 파일을 삭제하면 어떻게 될까?

쉘(Shell)

쉘이란 무엇인가?

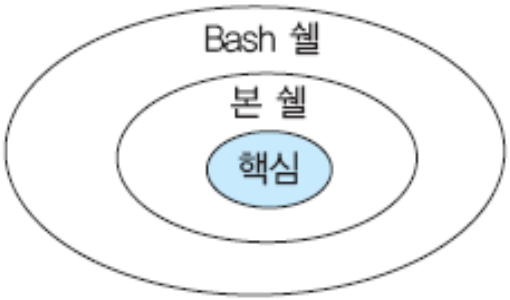
- 쉘의 역할
  - 쉘은 사용자와 운영체제 사이에 창구 역할을 하는 소프트웨어
  - 명령어 처리기(command processor)
  - 사용자로부터 명령어를 입력 받아 처리한다



쉘의 종류

- 유닉스/리눅스에서 사용 가능한 쉘의 종류

쉘의 종류	쉘 실행 파일
본 쉘	/bin/sh
콘 쉘	/bin/ksh
C 쉘	/bin/csh
Bash	/bin/bash
tcsh	/bin/tcsh



쉘의 종류

- 본 쉘(Bourne shell)
  - 벨연구소의 스티븐 본(Stephen Bourne)에 의해 개발됨
  - 유닉스에서 기본 쉘로 사용됨
- 콘 쉘(Korn shell)
  - 1980년대에는 역시 벨연구소에서 본 쉘을 확장해서 만듬.

쉘의 종류

- Bash(Bourne again shell)
  - GNU에서 본 쉘을 확장하여 개발한 쉘
  - 리눅스 및 맥 OS X에서 기본 쉘로 사용되면서 널리 보급됨
  - Bash 명령어의 구문은 본 쉘 명령어 구문을 확장함
- C 쉘(C shell)
  - 버클리대학의 빌 조이(Bill Joy)
  - 쉘의 핵심 기능 위에 C 언어의 특징을 많이 포함함
  - BSD 계열의 유닉스에서 많이 사용됨
  - 최근에 이를 개선한 tcsh이 개발되어 되어 사용됨

로그인 쉘(login shell)

- 로그인 하면 자동으로 실행되는 쉘
- 보통 시스템관리자가 계정을 만들 때 로그인 쉘 지정

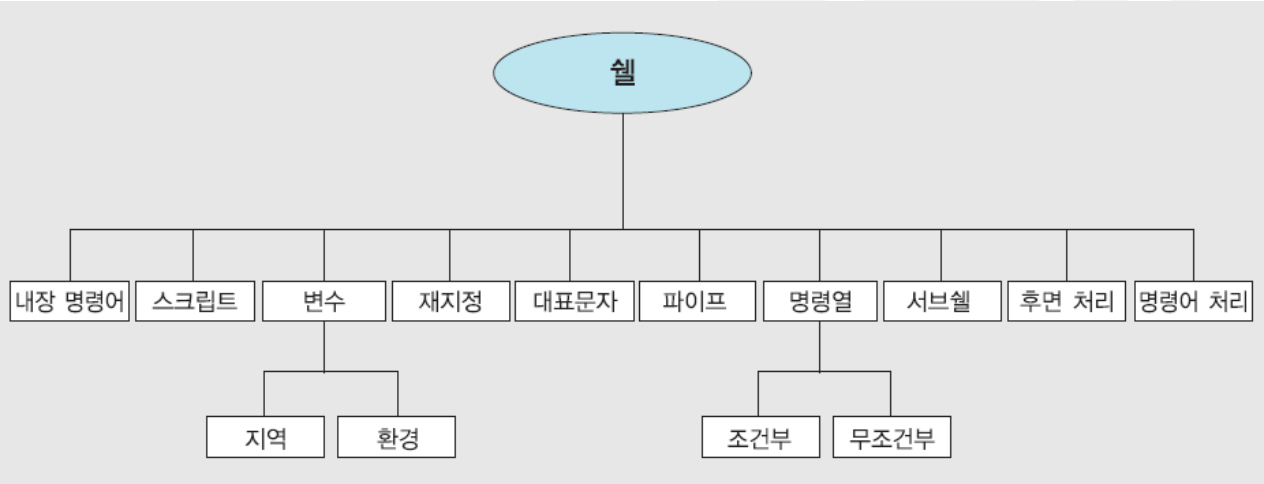
/etc/passwd

```
root:x:0:1:Super-User:/:/bin/bash
...
ywcho:x:109:101:ywcho:/user/faculty/ywcho:/bin/bash
```

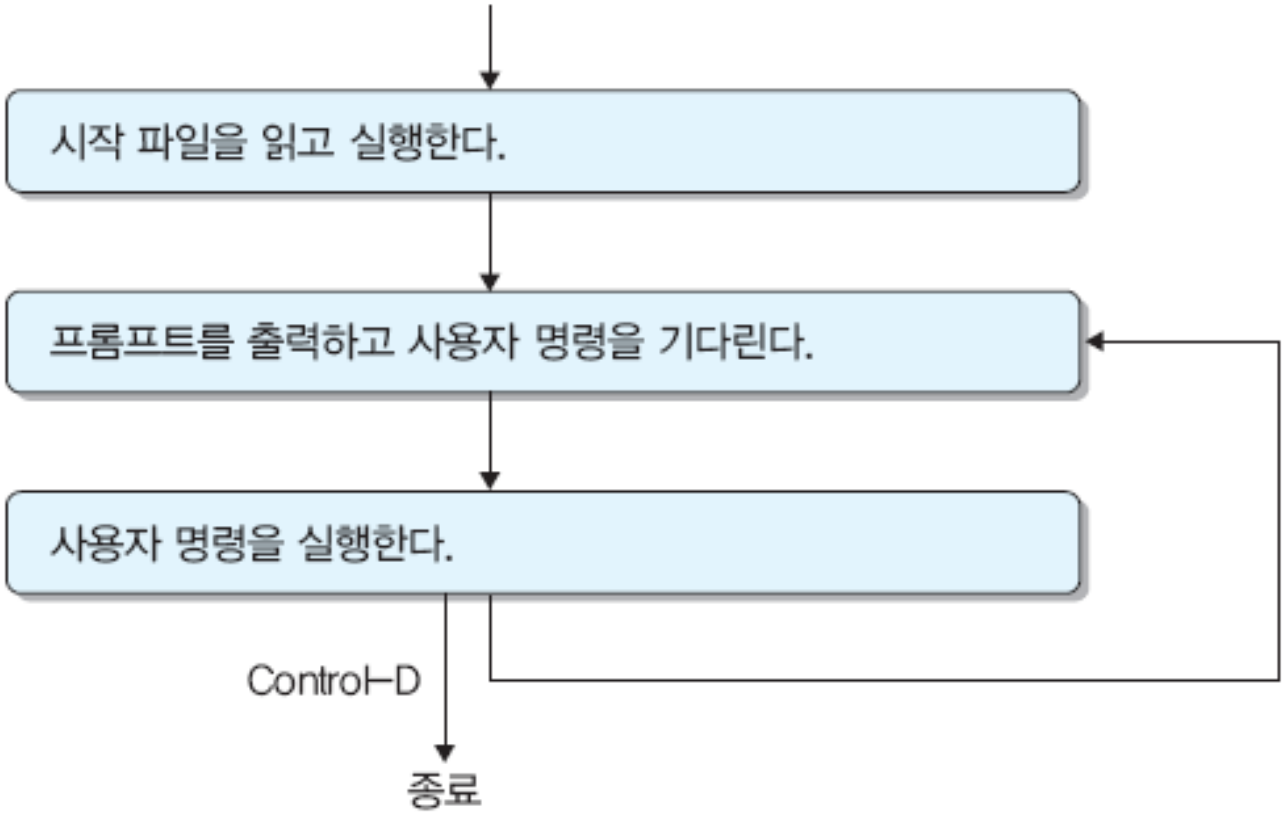
쉘의 기능

쉘의 주요 기능

- 명령어 처리
  - 사용자가 입력한 명령을 해석하고 적절한 프로그램을 실행
- 시작 파일
  - 로그인할 때 실행되어 사용자 별로 맞춤형 사용 환경 설정
- 스크립트
  - 쉘 자체 내의 프로그래밍 기능



쉘의 실행 절차


시작 파일을 읽고 실행한다.
프롬프트를 출력하고 사용자 명령을 기다린다.
사용자 명령을 실행한다.
Control-D
종료

쉘의 환경 변수

- 환경변수 설정법

> $ 환경변수명 = 문자열
>
> 환경변수의 값을 문자열로 설정한다.

```
$ TERM=xterm
$ echo $TERM
xterm
```

쉘의 환경 변수

- 환경변수 보기

```
$ env
TERM=xterm
SHELL=/bin/sh
GROUP=cs
USER=ywcho
HOME=/home/ywcho
PATH=/usr/local/bin:/usr/
bin: ...
...
```

- 사용자 정의 환경 변수

```
$ MESSAGE=hello
$ export MESSAGE
```

쉘의 시작 파일(start-up file)

- 시작 파일
  - 쉘마다 시작될 때 자동으로 실행되는 고유의 시작 파일
  - 주로 사용자 환경을 설정하는 역할을 하며
  - 환경설정을 위해서 환경변수에 적절한 값을 설정한다.
- 시스템 시작 파일
  - 시스템의 모든 사용자에게 적용되는 공통적인 설정
  - 환경변수 설정, 명령어 경로 설정, 환영 메시지 출력 등
- 사용자 시작 파일
  - 사용자 홈 디렉터리에 있으며 각 사용자에게 적용되는 설정
  - 환경변수 설정, 프롬프트 설정, 명령어 경로 설정, 명령어 이명 설정 등

쉘의 시작 파일(start-up file)

쉘의 종류	시작파일 종류	시작파일 이름	실행 시기
본 쉘	시스템 시작파일	/etc/profile	로그인
	사용자 시작파일	~/.profile	로그인
Bash 쉘	시스템 시작파일	/etc/profile	로그인
	사용자 시작파일	~/.bash_profile	로그인
	사용자 시작파일	~/.bashrc	로그인, 서브쉘
	시스템 시작파일	/etc/bashrc	로그인
C 쉘	시스템 시작파일	/etc/.login	로그인
	사용자 시작파일	~/.login	로그인
	사용자 시작파일	~/.cshrc	로그인, 서브쉘
	사용자 시작파일	~/.logout	로그아웃

쉘의 시작 파일(start-up file)

- profile

```
PATH=$PATH:/usr/local/bin:/etc
TERM=vt100
export PATH TERM
stty erase ^
```

- 시작 파일 바로 적용

```
$ . .profile
```