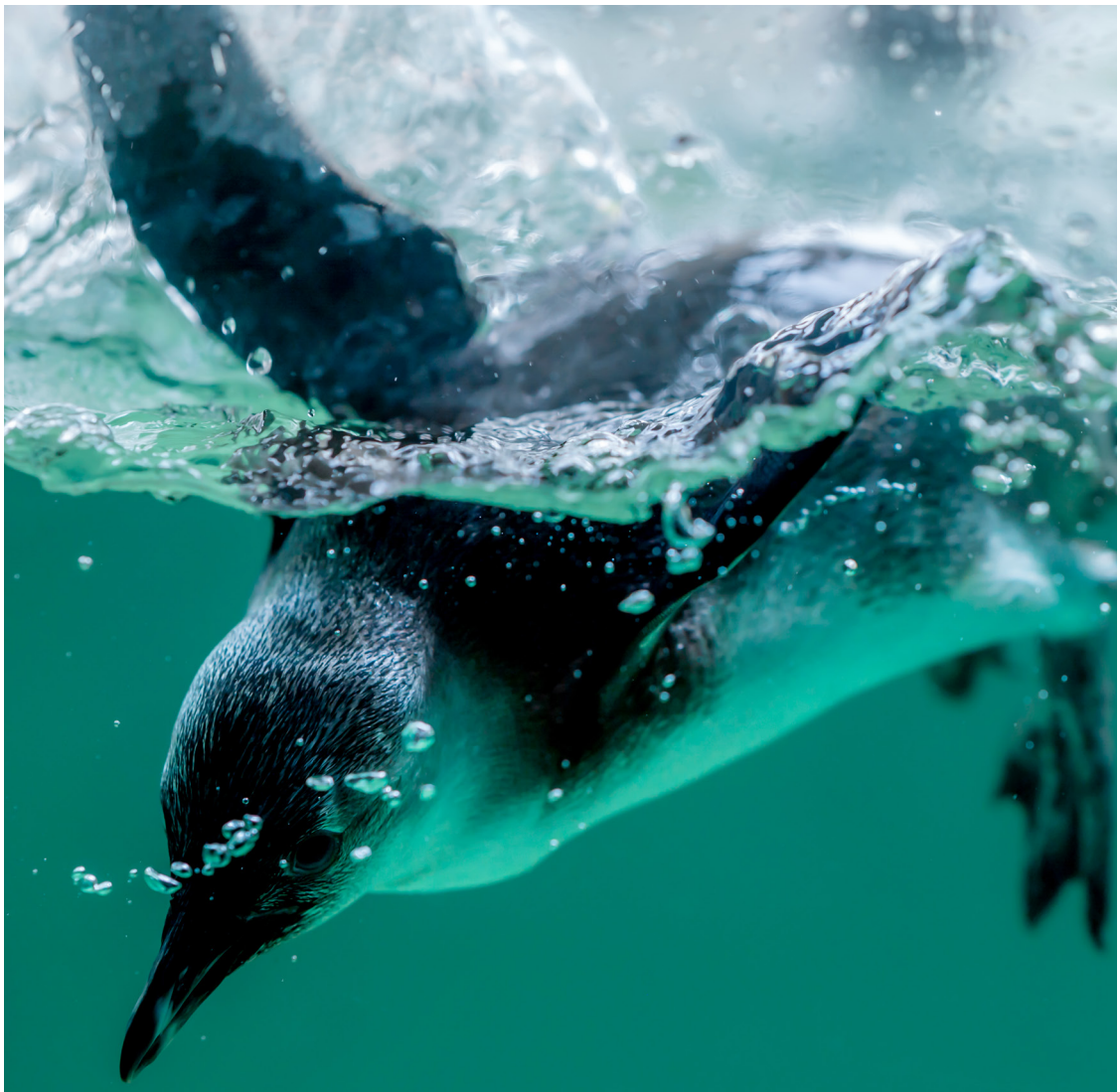
IDG
STRATEGIC
MARKETING SERVICES

IT WORLD CIO

IT WORLD HOW TO



기초부터 다시 시작하는 리눅스 커맨드 라인편

> 가장 기본적인 명령어
> 도움말 활용하기
> 명령어 치트 시트
> 파이프와 별칭, 스크립트의 이해와 활용
> 초보 탈출 지름길, 리눅스 트릭

기초부터 다시 시작하는 리눅스 커맨드 라인편

Sandra Henry-Stocker | Network World

Part 1. 가장 기본적인 명령어

리눅스 또는 유닉스에 입문하려면 먼저 리눅스가 어떻게 작동하는지 충분히 파악하고 커맨드 라인에서 사용하는 기본적인 명령어를 익히는 것이 좋다.

로그인

리눅스에 처음 로그인해서 터미널 창을 열거나 퍼티(PuTTY)와 같은 툴을 사용해 다른 시스템에서 리눅스 시스템에 로그인하면 홈 디렉터리에 위치하게 된다. 여기서 먼저 쓸만한 명령어는 다음과 같다.

```
pwd -- shows you where you are in the file system right now (stands for “present
working directory”)
whoami – confirms the account you just logged into
date -- show the current date and time
hostname -- display the system’s name
```

방금 사용자 이름과 비밀번호를 입력해 로그인했는데 whoami 명령어를 사용한다는 것이 언뜻 납득이 안 될 수 있다. 그러나 2개 이상의 계정을 사용하면 현재 사용 중인 계정을 확인하는 명령어를 알아 두는 것이 도움이 된다. 이 명령어의 실행 결과는 다음과 같다.

```
$ pwd
/home/justme
$ whoami
justme
$ date
Tue Nov 21 10:40:20 AM EST 2023$
$ hostname
venus
```

파일 나열하기

일반적으로 먼저 하는 작업 중 하나는 계정과 연결된 파일과 디렉터리를 확인하는 것이다. 다음 명령어는 현재

파일 시스템 위치에 있는 모든 파일과 디렉터리, 그리고 각 파일 및 디렉터리의 권한, 소유권까지 "긴 목록"으로 보여준다.

```
$ ls -al
total 16
drwx------. 1 justme justme   96 Nov  6 12:21 .
drwxr-xr-x. 1 root   root     56 Nov  6 12:15 ..
-rw-r--r--. 1 justme justme   18 Feb  5  2023 .bash_logout
-rw-r--r--. 1 justme justme  141 Feb  5  2023 .bash_profile
-rw-r--r--. 1 justme justme  526 Nov 11 12:21 .bashrc
drwxr-xr-x. 1 justme justme   34 Apr 13  2023 .mozilla
-rw-------. 1 justme justme 1445 Nov  6 12:21 .viminfo
-rw-r-----. 1 justme justme   35 Nov 21 12:34 myfile
```

출력된 내용을 보면 디렉터리는 3개("d"로 시작하는 줄), 파일은 5개("-"로 시작하는 줄)다. 세부 정보가 보이는 것은 ls 명령어에 -a(모두 표시)와 -l(세부 정보 표시) 옵션을 추가했기 때문이다. 참고로 소유자와 그룹은 모두 "justme"다. 기본적으로 사용자는 같은 이름의 사용자 그룹에만 속하는 멤버로, 시스템에 새 사용자가 추가될 때 설정된다. 날짜와 시간은 파일이 마지막으로 업데이트된 시점을 나타낸다.

파일 시스템 탐색하기

디렉터리 안으로 들어가려면 "cd .mozilla"와 같은 명령어를 사용하면 된다. 파일 시스템의 어디에 있든 현재 디렉터리는 "."으로 표현할 수 있다. 따라서 "cd ." 명령어는 아무런 작업도 수행하지 않는다. 이 명령어는 "지금 내가 있는 위치로 이동하라"는 뜻이기 때문이다. 반면 "cd .." 명령어를 사용하면 한 단계 위 디렉터리, 즉 파일 시스템의 시작 디렉터리인 /로 이동한다.

```
$ pwd
/home/justme
$ cd ..
$ pwd
/home
$ cd /tmp
$ pwd
/tmp
$ cd
$ pwd
/home/justme
```

"cd /tmp"와 같은 명령어를 사용하면 지정한 디렉터리로 이동하고, 인수 없는 cd 명령어는 항상 홈 디렉터리를 가리킨다. "cd ~", 또는 "cd /home/justme" 명령어를 사용해도 결과는 같다. 그러나 대부분 리눅스 사용자는 입력

을 최소화하는 편을 선호하므로 홈으로 돌아가려는 경우 간단히 cd만 입력한다.

참고로, 특정 디렉터리는 루트(superuser) 계정으로 전환하거나 su(switch user) 명령어를 사용해 적절한 권한을 가진 계정을 사용해야만 액세스할 수 있다. 액세스 권한이 없는 디렉터리로 이동하려 하면 다음과 같은 오류가 나타난다.

```
$ cd /home/newuser
-bash: cd: /home/newuser: Permission denied
```

액세스가 거부된 이유는 다음과 같이 디렉터리의 권한을 출력해 확인할 수 있다.

```
$ ls /home/newuser
drwx------. 1 newuser newuser   80 Jul 24 10:48 newuser
```

이 디렉터리에 대한 액세스 권한을 가진 유일한 계정이 “newuser”라는 것을 알 수 있다. 참고로 시스템 관리를 담당하는 경우를 제외하면 대부분 su 명령어를 사용하도록 설정된다.

파일 및 디렉터리 권한 이해

액세스할 수 있는 항목과 없는 항목을 파악하려면 파일 및 디렉터리의 권한이 작동하는 방식을 이해하는 것이 중요하다. "drwxr—"의 "d"와 같은 권한 문자열의 첫 번째 문자는 파일 형식을 나타낸다. 디렉터리 목록에 있는 대부분 항목은 다음 중 하나다.

```
d = directory
- = file
```

권한 문자열의 나머지 부분은 파일/디렉터리 소유자, 소유자의 기본 그룹, 그 외 모든 사용자에게 주어진 권한을 나타낸다. 권한 문자열을 그룹으로 나누면 다음과 같다.

```
d rwx r-- ---
^  ^   ^   ^
|  |   |   |
|  |   |   +-- everyone else’s permissions (no access)
|  |   +------ other group members’ permissions (read only)
|  +---------- owner’s permissions (read, write and execute)
+------------- file type (d = directory)
```

이 예시의 "rwx"는 디렉터리 소유자에게 읽기(r), 쓰기(w), 실행(x) 권한이 부여됐음을 의미한다. 반면 "-"는 이런 권한이 제공되지 않았다는 뜻이다. 보통은 읽기만(r-) 또는 읽기 및 실행(r-x) 권한만 부여한다. 이런 방식으로 파일에 액세스할 수 있는 사용자를 제어할 수 있다. 예를 들어 사용자에게 홈 디렉터리에 대한 액세스 권한도 있다

면, 다음과 같은 명령어로 디렉터리의 파일에 대한 액세스 권한을 부여할 수 있다.

```
$ chmod o+r myfile
```

이 명령어는 "other" 그룹의 모든 사용자에게, 즉 소유자 또는 파일의 할당 그룹 구성원이 아닌 모든 사용자에게 읽기 액세스 권한을 부여한다. 유의할 점은 이런 사용자의 경우 해당 파일이 포함된 디렉터리에 대한 권한이 있어야 액세스가 가능하다는 것이다. 참고로, 짐작하겠지만 파일의 권한은 파일 소유자 또는 루트만 변경할 수 있다.

파일 내용 보기

파일 내용 보기는 파일 형식에 따라 다르다. 텍스트 파일을 보려면 cat(파일 내용 표시) 또는 more(한 번에 한 화면 분량의 파일 내용 표시) 명령어를 사용하면 된다. 배시(Bash) 스크립트는 텍스트 파일이며 .bashrc와 같은 시작 파일도 마찬가지다. 반면 시스템 실행 파일은 "바이너리" 파일이라고 하는데, 이런 파일에 일반적인 보기 명령어를 사용하면 내용이 긴 0과 1, 즉 바이너리로 표시된다. 이미지와 같은 콘텐츠를 데스크톱에 표시할 수 있는 명령어는 또 다른 범주에 속한다.

```
$ cat myfile
This is my favorite file. It reminds me about how I felt when I first
Discovered Linux!
$ head -5 colors
aqua
azure
black
blue
bronze
$ tail -5 colors
turquoise
violet
wheat
white
yellow
$ more colors
aqua
azure
black
blue
bronze
brown
chocolate
…
```

more 명령어는 한 번에 한 화면만큼만 파일 내용을 나열한다. Enter 키를 누르면 다음 화면으로 넘어간다.

파일 복사, 이동 및 삭제

파일 복사본을 만들려면 다음과 같이 cp(copy) 명령어를 사용한다.

```
$ cp myfile myfile.save
```

파일을 쓰기 액세스 권한이 있는 다른 디렉터리로 옮기려면 다음과 같이 하면 된다.

```
$ mv myscript ~/bin
```

이 명령어는 "myscript"라는 파일을 홈 디렉터리의 bin 디렉터리로 복사하는 내용이다. bin 디렉터리가 없는 경우 "mkdir ~/bin" 명령어를 사용해서 만든다. 파일을 삭제하려면 rm 명령어를 사용한다. 당연히 파일에 대한 쓰기 권한이 필요하다.

```
$ rm myfile
```

Part 2. 도움말 활용하기

리눅스를 꽤 오랫동안 사용했다 해도 새로운 명령어를 익히거나 명령어의 수많은 옵션에 대한 정보를 얻기 위해 도움말을 참고해야 할 때가 있다. 시스템 자체에서 얻을 수 있는 몇 가지 도움말에 대해 알아보자.

도움말을 제공하는 명령어

리눅스에 대한 도움말을 얻는 가장 유용한 명령어는 man(manual을 의미)이다. man은 명령어가 무슨 일을 하며 어떤 옵션이 있는지에 대한 정보를 제공한다. 거의 모든 명령어에는 man 페이지가 있으므로 이 페이지를 사용해 한 번에 한 화면씩 정보를 볼 수 있다. man 페이지를 보는 방법은 간단하다. "man ls" 또는 "man date"와 같이 사용하면 된다. 예를 들어 다음 명령어는 date의 man 페이지를 보여준다. 'head -11' 부분은 명령어 출력에서 표시되는 내용을 11줄로 제한한다. 여기서 man 페이지 전체를 포함할 필요는 없기 때문이다.

```
$ man date | head -11
DATE(1)                          User Commands                          DATE(1)
```

```
NAME
       date - print or set the system date and time

SYNOPSIS
       date [OPTION]... [+FORMAT]
       date [-u|--utc|--universal] [MMDDhhmm[[CC]YY][.ss]]

DESCRIPTION
       Display date and time in the given FORMAT.  With -s, or with [MMDDhhmm[[CC]YY]
[.ss]], set the date and time.
```

또 다른 유용한 명령어는 이름도 간단한 "help"다. 이 명령어는 배시 내장 명령어, 즉 배시 실행 파일에 내장된 명령어에 대한 정보를 제공하며, 따라서 자체 man 페이지는 없다. 다음 명령어는 date 명령어에 대해 아무런 정보도 제공하지 않는다. 내장 명령어가 아니기 때문이다. 가끔 이 같은 화면을 보더라도 놀랄 필요는 없다.

```
$ help date
-bash: help: no help topics match `date'.  Try `help help' or `man -k date' or `info
date'.
```

"help help"라고 입력하면 다음과 같은 화면이 나온다.

```
$ help help
help: help [-dms] [pattern ...]
    Display information about builtin commands.

    Displays brief summaries of builtin commands.  If PATTERN is
    specified, gives detailed help on all commands matching PATTERN,
    otherwise the list of help topics is printed.

    Options:
      -d        output short description for each topic
      -m        display usage in pseudo-manpage format
      -s        output only a short usage synopsis for each topic matching
                PATTERN

    Arguments:
      PATTERN   Pattern specifying a help topic

    Exit Status:
    Returns success unless PATTERN is not found or an invalid option is given.
```

배시 내장 명령어 목록을 보려면 help와 같은 명령어를 단독으로 사용하면 된다. 참고로 이 명령어는 상당히 긴 결과를 반환하므로 리눅스 초보자에게는 혼란스러울 수 있다. 다음 화면에는 처음 몇 줄만 나와 있는데, 이 뒤에 각 내장 명령어의 목록과 설명이 나온다.

```
$ help | head -7
GNU bash, version 5.2.15(1)-release (x86_64-redhat-linux-gnu)
These shell commands are defined internally.  Type `help' to see this list.
Type `help name' to find out more about the function `name'.
Use `info bash' to find out more about the shell in general.
Use `man -k' or `info' to find out more about commands not in this list.

A star (*) next to a name means that the command is disabled.
```

다음 출력을 보면 help 명령어 자체가 배시 내장 명령어임을 명확히 알 수 있다.

```
$ help | grep help
These shell commands are defined internally.  Type `help' to see this list.
Type `help name' to find out more about the function `name'.
 help [-dms] [pattern ...]   <=== it's in the builtins list
```

전체 출력은 40줄 이상이지만 다음과 같이 이 출력을 more 명령어로 전달하면 한 번에 한 화면씩 볼 수 있다.

```
$ help | more
```

출력의 나머지 부분은 다음과 같다.

```
alias [-p] [name[=value] ... ]                    logout [n]
```

man -k 명령어를 사용해 검색어와 관련된 명령어의 짧은 설명을 볼 수도 있다. 다음은 하나의 설명만 있는 예시지만 대부분 검색어는 이보다 훨씬 더 설명이 많다.

```
$ man -k calendar
cal (1)              - display a calendar
```

echo 명령어 사용

echo 명령어는 입력한 내용을 반복한다. 문자열을 사용하면 단순히 그 문자열을 표시하지만 변수를 표시하도록 지시하면 변수의 값을 표시한다. 다음 두 번째 명령어에서 $SHELL 변수의 값은 사용 중인 셸이다.

```
$ echo hello!
```

```
hello!
$ echo $SHELL
/bin/bash
```

/etc/passwd 파일의 항목을 통해 현재 사용 중인 셸을 확인할 수도 있다. 계정을 설명하는, 콜론으로 구분된 줄의 마지막 필드다. 숫자로 된 사용자 ID와 그룹 ID도 볼 수 있다. 다음 예제에서는 둘 다 1004다.

```
$ grep justme /etc/passwd
justme:x:1004:1004:Just Me:/home/justme:/bin/bash
```

명령어 출력을 파일로 저장하기

명령어의 출력을 파일로 저장하려면 > 기호를 사용해 출력을 리디렉션한다. 다음 예에서 date 명령어의 출력은 "times"라는 파일로 전송되며, 이 파일이 이미 있는 경우에는 덮어쓴다. 두 번째 명령어의 출력은 파일에 추가된다.

```
$ date > times
$ date >> times
$ cat times
Wed Nov  8 03:15:18 PM EST 2023
Wed Nov  8 03:15:24 PM EST 2023
```

Part 3. 리눅스 명령어 치트 시트

리눅스 고수로 가는 데 도움이 되는 것이 커맨드 라인에서 해야 할 모든 일을 알려 주는 명령어 용어다. 이 명령어 용어를 익히는 가장 좋은 방법은 일반적으로 사용되는 명령어와 간단한 설명, 각 명령어 활용 예시가 정리된 목록인 "치트 시트"부터 시작하는 것이다. 지금까지 있는지도 미처 몰랐던 명령어를 치트 시트를 통해 익힐 수 있다.

다음은 리눅스 입문용 치트 시트다. 일련의 명령어와 명령어에 대한 설명, 예시가 포함되며, 주 기능 그룹별로 분류했다. 참고로 대부분 리눅스 명령어는 다양한 배포판에서 똑같이 작동한다. 예를 들어 툴, 프로그래밍 언어 등 관리 패키지를 설치하는 데 사용하는 명령어가 대표적이다. 반면 배포판마다 다른 명령어도 있다.

간단한 치트 시트

다음 표에는 가장 기본적인 리눅스 명령어에 대한 정보가 나와 있다.

계정 관련 명령어

명령어	기능	예제
pwd	현재 파일 시스템 위치 출력	pwd
whoami	username 출력	whoami
who	로그인한 사용자 출력	who
ls	현재 위치의 파일 출력	ls; ls -l; ls -a
env	검색 경로, 명령어 히스토리 등 설정 출력	env
echo	기존 인수 반복	echo hello; echo $PATH
history	최근에 사용한 명령어 리스트	history; history | tail -5
passwd	암호 초기화	passwd

텍스트 파일 내용 보기

명령어	기능	예제
cat	파일 내용 출력	cat myfile
more	파일 내용 한 페이지씩 출력	more myfile
less	파일 내용 한 페이지씩 출력, 돌아가기 가능	less myfile
head	파일 시작 내용 출력	head myfile; head -5 myfile
tail	파일 마지막 내용 출력	tail myfile; tail -3 myfile

파일 관리

명령어	기능	예제
chmod	파일 권한 변경	chmod a+x myscript; chmod 700 myscript
cp	파일 복사본 만들기	cp myfile savefile
mv	파일 이름 바꾸거나 다른 위치로 이동	mv file oldfile; mv script bin
rm	파일 삭제	rm oldfile

실행 중인 프로세스 보기

명령어	기능	예제
ps	현재 로그인 관련 프로세스 출력	ps
ps -ef	시스템에서 실행 중인 프로세스 출력	ps -ef; ps -ef | more

명령어 출력

앞서 표로 설명한 다양한 명령어의 샘플 출력을 살펴보자.

env

env 명령어는 명령어 환경에 대해 많은 정보를 제공한다. 계정에 할당된 셸, 명령어 실행 기록에 유지되는 명령어의 수(HISTSIZE), 기본 텍스트 편집기, 현재 파일 시스템 위치, 터미널 유형 및 이전 파일 시스템 위치 등 여러 데이터를 볼 수 있다.

```
$ env
SHELL=/bin/bash
HISTCONTROL=ignoredups
HISTSIZE=1000
HOSTNAME=fedora
EDITOR=/usr/bin/nano
PWD=/home/justme
LOGNAME=justme
HOME=/home/justme
LANG=en_US.UTF-8
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33
;01:cd=40;33;01:or=40;31;01:mi=01;37;41:su=37;41:sg=30;43:ca=00:tw=30;42:
ow=34;42:st=37;44:ex=01;32:*.tar=01;31:*.tgz=01;31:*.arc=01;31:*.arj=01;31:*.
taz=01;31:*.lha=01;31:*.lz4=01;31:*.lzh=01;31:*.lzma=01;31:*.tlz=01;31:*.
txz=01;31:*.tzo=01;31:*.t7z=01;31:*.zip=01;31:*.z=01;31:*.dz=01;31:*.gz=01;31:*.
lrz=01;31:*.lz=01;31:*.lzo=01;31:*.xz=01;31:*.zst=01;31:*.tzst=01;31:*.bz2=01;31:*.
bz=01;31:*.tbz=01;31:*.tbz2=01;31:*.tz=01;31:*.deb=01;31:*.rpm=01;31:*.jar=01;31:*.
war=01;31:*.ear=01;31:*.sar=01;31:*.rar=01;31:*.alz=01;31:*.ace=01;31:*.zoo=01;31:*.
cpio=01;31:*.7z=01;31:*.rz=01;31:*.cab=01;31:*.wim=01;31:*.swm=01;31:*.dwm=01;31:*.
esd=01;31:*.avif=01;35:*.jpg=01;35:*.jpeg=01;35:*.mjpg=01;35:*.mjpeg=01;35:*.
gif=01;35:*.bmp=01;35:*.pbm=01;35:*.pgm=01;35:*.ppm=01;35:*.tga=01;35:*.xbm=01;35:*.
xpm=01;35:*.tif=01;35:*.tiff=01;35:*.png=01;35:*.svg=01;35:*.svgz=01;35:*.mng=01;35:*.
pcx=01;35:*.mov=01;35:*.mpg=01;35:*.mpeg=01;35:*.m2v=01;35:*.mkv=01;35:*.
webm=01;35:*.webp=01;35:*.ogm=01;35:*.mp4=01;35:*.m4v=01;35:*.mp4v=01;35:*.
vob=01;35:*.qt=01;35:*.nuv=01;35:*.wmv=01;35:*.asf=01;35:*.rm=01;35:*.rmvb=01;35:*.
flc=01;35:*.avi=01;35:*.fli=01;35:*.flv=01;35:*.gl=01;35:*.dl=01;35:*.xcf=01;35:*.
xwd=01;35:*.yuv=01;35:*.cgm=01;35:*.emf=01;35:*.ogv=01;35:*.ogx=01;35:*.aac=01;36:*.
au=01;36:*.flac=01;36:*.m4a=01;36:*.mid=01;36:*.midi=01;36:*.mka=01;36:*.mp3=01;36:*.
mpc=01;36:*.ogg=01;36:*.ra=01;36:*.wav=01;36:*.oga=01;36:*.opus=01;36:*.spx=01;36:*.
xspf=01;36:*~=00;90:*#=00;90:*.bak=00;90:*.old=00;90:*.orig=00;90:*.part=00;90:*.
rej=00;90:*.swp=00;90:*.tmp=00;90:*.dpkg-dist=00;90:*.dpkg-old=00;90:*.ucf-
dist=00;90:*.ucf-new=00;90:*.ucf-old=00;90:*.rpmnew=00;90:*.rpmorig=00;90:*.
rpmsave=00;90:
```

```
TERM=xterm
LESSOPEN=||/usr/bin/lesspipe.sh %s
USER=justme
SHLVL=1
DEBUGINFOD_URLS=https://debuginfod.fedoraproject.org/
XDG_DATA_DIRS=/home/justme/.local/share/flatpak/exports/share:/var/lib/flatpak/exports/
share:/usr/local/share:/usr/share
PATH=/home/justme/.local/bin:/home/justme/bin:/usr/local/bin:/usr/bin:/usr/local/
sbin:/usr/sbin
MAIL=/var/spool/mail/justme
_=/usr/bin/env
OLDPWD=/tmp
```

echo

echo 명령어는 사용자가 제공한 내용과 포함한 변수를 표시하지만 기존 파일에 추가하는 데도 사용할 수 있다.

```
$ echo date: `date`
date: Tue Nov 20 02:53:43 PM EST 2023
```

참고로 단순히 이름을 표시하는 것이 아니라 echo 명령어에 명령어 출력을 포함하려면 백틱(`)을 사용해 명령어를 실행해야 한다. 생성된 출력을 파일에 추가하려면 다음과 같이 하면 된다.

```
$ echo date: `date` > myreport
$ cat myreport
date: Tue Nov 20 03:06:33 PM EST 2023
```

>는 이전 파일 내용을 덮어쓰므로 기존 파일에 추가하는 경우에는 >>를 사용해야 한다.

history

history 명령어는 $HISTSIZE 설정에 허용된 한도 내에서 최근 실행한 명령어를 보여준다. 다음과 같은 명령어를 사용하면 최근 4개 명령어를 볼 수 있다.

```
$ history | tail -3
   84  echo date: `date` > myreport
   85  cat myreport
   86  history | tail -3
```

passwd 명령어를 사용하면 비밀번호를 바꿀 수 있지만, 이를 위해서는 현재 비밀번호를 먼저 입력해야 한다. 또한 비밀번호의 길이와 복잡성도 일정 조건을 충족해야 한다.

```
$ passwd
```

```
Changing password for user justme.
Current password:
New password

Retype new password:
passwd: all authentication tokens updated successfully.
```

more

more 명령어는 한 번에 한 화면씩 파일 내용을 표시한다. 모든 내용을 다 볼 필요는 없으므로, 여기서는 -n 옵션을 사용해 5줄만 표시했다. 이 화면에서 Enter 키를 누르면 다음 5줄을 볼 수 있다.

```
$ more -n 5 TZs
Africa/Abidjan
Africa/Accra
Africa/Addis_Ababa
Africa/Algiers
Africa/Asmara
```

head와 tail

head와 tail 명령어는 텍스트 파일의 맨 위 줄과 맨 아래 줄을 표시한다. more 명령어와 마찬가지로 -n 옵션을 추가해 화면에 표시하는 줄 수를 제한할 수 있다.

```
$ head -n 5 TZs
Africa/Abidjan
Africa/Accra
Africa/Addis_Ababa
Africa/Algiers
Africa/Asmara
$ tail -n 5 TZs
UTC
Universal
W-SU
WET
Zulu
```

chmod

chmod 명령어는 파일 권한을 변경하는 데 사용한다. 다른 사람들이 실행할 수 있는 스크립트를 설정하려면 다음과 같이 실행 권한을 부여하면 된다.

```
$ chmod 755 myscript
```

이 chmod에서 첫 번째 5는 같은 그룹의 다른 사용자가 스크립트를 읽고 실행할 수 있도록 권한을 설정한다. 두 번째 5는 다른 모든 사용자가 스크립트를 실행하도록 허용한다. 파일 위치에 대한 액세스 권한이 필요하다.

```
$ chmod 755 mkTable
$ ls -l mkTable
-rwxr-xr-x. 1 justme justme 701 Nov 17 14:05 mkTable
```

cp

cp 명령어는 파일 복사본을 만든다. 문제가 발생할 경우 손쉽게 롤백할 수 있도록 파일을 변경하기 전에 백업 복사본을 만들어 두는 것이 좋다.

```
$ cp myfile myfile.backup
```

mv와 rm

mv(move) 명령어를 이용하면 파일 이름을 변경하고 다른 위치로 이동할 수 있다. 예를 들면 다음과 같다.

```
$ mv myreport oldreport
$ mv myscript ~/bin
$ mv myreport /shared/reports
```

rm(remove) 명령어로 파일을 삭제할 수도 있다.

```
$ rm oldfile
$ ls oldfile
ls: cannot access 'oldfile': No such file or directory
```

마지막 ls 명령어를 보면 rm 명령어가 실행된 후 더 이상 파일을 사용할 수 없음을 확인할 수 있다.

ps

ps 명령어는 실행 중인 프로세스를 표시한다. 인수가 없으면 현재 로그인과 관련된 프로세스만 보여준다.

```
$ ps
    PID TTY          TIME CMD
   6688 pts/1    00:00:00 bash
   7201 pts/1    00:00:00 ps
```

ps -ef(일부 시스템에서는 ps -aux)를 사용하면 실행 중인 모든 프로세스를 볼 수 있다. 출력을 more 또는 grep으로 보내서 찾는 항목만 출력되도록 할 수 있다.

```
$ ps -ef | grep sshd
root         782       1  0 Nov20 ?        00:00:00 sshd: /usr/sbin/sshd -D
root        6632     782  0 14:38 ?        00:00:00 sshd: shs [priv]
justme      6678    6632  0 14:38 ?        00:00:00 sshd: shs@pts/1
justme      7228    6688  0 16:14 pts/1    00:00:00 grep --color=auto sshd
```

예제 명령어 출력은 sshd(보안 로그인)와 관련된 프로세스를 보여준다.

who

who 명령어는 시스템 콘솔이든 원격 로그온이든 관계없이 현재 시스템에 로그인한 모든 사용자의 목록을 보여준다.

```
$ who
fedora   seat0        2023-11-16 09:06 (login screen)
fedora   tty2         2023-11-16 09:06 (tty2)
justme   pts/1        2023-11-17 14:38 (192.168.0.8)
```

리눅스 커맨드 라인을 편안하게 느끼는 수준이 되려면 리눅스에 대해 배워야 할 것이 많지만, 그만한 가치는 충분하다.

Part 4. 파이프와 별칭, 스크립트의 이해와 활용

리눅스 커맨드 라인에서 실행할 수 있는 가장 다재다능한 것은 '파이프(pipe)'를 사용해 한 명령어의 출력 결과를 다른 명령어로 전달하는 기능이다. 또한 복잡한 명령어를 '별칭(aliases)'으로 설정해 간단한 문자열을 입력하는 방법으로 실행할 수도 있다.

파이프 사용하기

커맨드 라인 작업과 관련해 필자가 예전부터 좋아하는 기능 중 하나는 세로 막대(|)로 표현되는 파이프를 사용해 한 명령어의 출력을 다른 명령어로 전달한다. 예를 들어 다음과 같은 명령어로 리눅스 서버에 로그인한 사람의 수를 계산할 수 있다.

```
$ who | wc -l
3
```

여기서 who 명령어는 로그인한 사용자를 보여준다. 그리고 wc -l 명령어는 who 명령어가 보낸 출력의 줄 수를 계산한다. 다음과 같이 최근에 실행한 명령어를 볼 수 있다.

```
$ history | tail -3
 1014  ls bin
 1015  uptime
 1016  history | tail -3
```

현재 디렉터리에서 가장 최근에 업데이트된 파일을 보려면 다음과 같이 하면 된다.

```
$ ls -ltr | tail -3
-rw-r--r--. 1 justme justme   2459 Nov 17 13:59 AllTables
-rwx--x--x. 1 justme justme    701 Nov 17 14:05 mkTable
-rwxr-x---. 1 justme justme     86 Nov 20 11:25 myscript
```

별칭 만들기

리눅스를 사용하며 시간을 절약하는 가장 효과적인 방법은 복잡한 명령어를 별칭으로 바꾸는 것이다. 즉, 단어 또는 "ll"과 같은 약어를 "ls -l"와 같이 명령어에 연결하거나 "recent"와 같은 단어를 "history | tail -10"과 같은 명령어에 연결해 최근에 사용한 10가지 명령어를 볼 수 있다.

별칭을 사용하면 일일이 입력해야 하는 명령어를 줄여 시간을 절약하고 명령어의 모든 세부 항목을 제공해 입력 실수를 방지한다. 자주 사용할 때는 입력하는 양이 줄어들고, 가끔 사용할 때는 사용 방법을 힘들게 기억하지 않아도 된다는 점에서 장점이 많다.

별칭을 저장하려면 .bashrc 파일에 명령어를 넣으면 된다. 저장하면 터미널 창을 열어 항상 사용할 수 있다.

```
alias myprocs="ps -ef | grep `whoami`"  # show my processes
alias c="clear"                         # clear the screen
alias rec="ls -ltr | tail -3"           # show recent file updates
```

검색 경로 $PATH에 대한 이해

리눅스 시스템의 검색 경로는 콜론(:)으로 구분된 파일 시스템 위치의 시퀀스로, 명령어를 입력할 때 해당 명령어를 찾는 데 사용한다. 예를 들어 "date"를 입력하면 셸은 해당 이름의 실행 파일을 찾을 때까지 검색 경로의 각 위치를 탐색하고, 명령어를 찾으면 실행한다. PATH 설정에는 "/usr/bin" 및 "/usr/local/bin"과 같은 디렉터리가 포함돼야 한다. 다음과 같이 검색 경로를 살펴볼 수 있다.

```
$ echo $PATH
/home/justme/bin:/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin
```

텍스트 파일 편집

노트, 목록, 스크립트 등 텍스트 파일을 생성하려면 먼저 리눅스 텍스트 편집기 중 하나를 익혀야 한다. 현재 많은 리눅스 시스템에서 기본 편집기는 'vim'이다. vim을 사용하면 손쉽게 내용을 추가하고 파일을 탐색하고 단어 또는 줄을 찾아 바꿀 수 있다.

스크립트는 별칭과 마찬가지로 명령어의 이름을 입력하는 간단한 방법으로 일련의 명령어를 실행할 수 있게 해준다. 아주 간단한 스크립트의 예를 들면 다음과 같다.

```
#!/bin/bash

echo This script is running in $SHELL
echo The user running it is $USER
```

여기서 첫 번째 줄은 스크립트를 실행하는 데 사용할 셸을 식별한다. 이 줄이 없으면 현재 사용 중인 셸이 스크립트를 실행하게 된다. 다른 두 줄은 사용 중인 스크립트와 이를 실행하는 사용자의 이름을 표시한다. 이와 같은 간단한 스크립트를 만들려면 "vim my script"와 같은 명령어로 vim을 시작한다. "i"를 눌러 입력 모드로 들어간 다음 텍스트를 입력한다. 텍스트 입력을 마치면 Enter 키를 눌러 입력 모드에서 나온 다음 ":wq"를 눌러 파일을 저장하고 vim을 종료한다.

참고로 스크립트 이름을 입력해 실행하려면 다음와 같이 스크립트에 실행 권한을 부여해야 한다.

```
$ chmod 750 myscript
$ ./myscript
This script is running in bash
The user running it is justme
```

이 예에서는 현재 파일 시스템 위치가 검색 경로에 포함되지 않을 가능성이 높으므로 스크립트 이름 앞에 "./"를 붙였다.

이처럼 스크립트를 이용하면 많은 작업을 더 쉽게 실행할 수 있으며, 특히 복잡한 명령어가 포함된 경우 효과가 크다. 일련의 값으로 명령어를 실행할 수 있는 루프, 또는 일부 변수에 따라 다르게 실행되는 case문 등이 대표적이다.

Part 5. 초보 탈출 지름길, 리눅스 트릭

리눅스 커맨드 라인을 사용하면 많은 작업을 할 수 있으며 특히 "트릭"을 익히면 더 생산적으로 작업할 수 있다.

리눅스에 입문할 때 첫 번째 과제는 ls(파일 나열), pwd(현재 작업 중인 디렉터리 표시), cat(텍스트 파일의 내용 표시), cd(다른 디렉터리로 이동) 같은 명령어를 익히는 것이다. 이와 같은 명령어를 비롯해 다른 많은 기본적인 명령어를 익혔다면 대부분의 리눅스 애용자가 말하는 "리눅스 트릭(Linux trick)"을 배울 차례다.

"리눅스 트릭"이란 리눅스 커맨드 라인을 더 쉽게 사용하기 위한 다양한 방법을 의미한다. 특히 복잡한 프로세스를 빠르고 편리하게 만들고 싶을 때 사용한다. 가령 파일을 그냥 나열하는 것이 아니라 크기별로 정렬하거나 특정 문구가 포함된 파일만 나열하려면 어떻게 해야 할까? 흔히 사용하는 명령어를 실행하기는 쉽지만 최근에 사용한 명령어에서 중요한 변경 사항을 적용해 다시 실행하려는 경우는 어떨까? 이런 상황에서 트릭은 복잡한 명령어를 더 쉽게 사용할 수 있는 방법이다.

주석 인식

우선 명령어의 주석부터 살펴보자. 주석은 #(파운드 기호) 뒤에 나오는 내용으로, 셸에서 명령어로 인식해 실행하는 일이 없도록 방지하면서 명령어와 관련된 맥락 정보를 제공한다. 예를 들어 다음은 커맨드 라인에서 직접 입력하든 스크립트에 포함되든, 어떤 작업을 하는지 설명하는 주석에서 흔히 볼 수 있다.

```
$ # This is a comment
```

이는 명령어 프롬프트에 입력된, 부가적인 명령어 없이 명확한 주석을 보여준다. 이처럼 일반적으로 주석은 코드의 다양한 부분을 더 직관적으로 이해하고, 필요할 때 더 쉽게 업데이트할 수 있도록 스크립트에 포함된다.

자동 완성

자동 완성을 위해서는 파일 또는 명령어 이름 중간에서 tab 키를 누른다. 그러면 셸이 나머지 부분을 자동으로 입력해준다. 둘 이상의 일치 항목이 발견되는 경우 더 많은 문자가 입력될 때까지 기다린다.

```
$ ls ha^tab                    # type “ls” followed by a tab
happy_quotes
```

텍스트 파일의 일부분 보기

head, tail, more 명령어를 사용하면 파일의 특정 부분을 볼 수 있다. 몇 가지 예를 들면 다음과 같다.

```
$ head -2 happy_quotes
You're never fully dressed without a smile.
Let a smile be your umbrella.
$ tail -2 happy_quotes
Happiness depends upon ourselves.
Happiness is when what you think, what you say, and what you do are in harmony.
$ more states
Alabama
Alaska
Arizona
Arkansas
California
Colorado
Connecticut
Delaware
Florida
Georgia
Hawaii
Idaho
--More--(21%)
```

column 명령어도 알아두면 좋다. 미국의 모든 주를 한꺼번에 표시할 때 주 목록을 여러 페이지에 걸쳐 표시하지 않고 한 화면에서 볼 수 있다.

```
$ more states | column
Alabama         Hawaii          Massachusetts   New Mexico      South Dakota
Alaska          Idaho           Michigan        New York        Tennessee
Arizona         Illinois        Minnesota       North Carolina  Texas
Arkansas        Indiana         Mississippi     North Dakota    Utah
California      Iowa            Missouri        Ohio            Vermont
Colorado        Kansas          Montana         Oklahoma        Virginia
Connecticut     Kentucky        Nebraska        Oregon          Washington
Delaware        Louisiana       Nevada          Pennsylvania    West Virginia
Florida         Maine           New Hampshire   Rhode Island    Wisconsin
Georgia         Maryland        New Jersey      South Carolina  Wyoming
```

하나의 긴 주 목록이 포함된 파일을 줄당 5개 주가 포함된 파일로 변환하려면 다음과 같이 하면 된다.

```
$ cat states  | column > states-columns
```

열의 개수는 화면 너비에 따라 달라진다.

명령어 기록 보기

history를 사용하면 최근에 실행한 명령어를 볼 수 있다. 기억할 수 있는 명령어의 수는 HISTSIZE 설정에 따라 결정된다. 다음과 같이 이 설정의 값을 확인할 수 있다.

```
$ echo $HISTSIZE
1000
```

이전에 실행한 명령어를 보려면 history를 실행한다. 몇 가지 예를 들면 다음과 같다.

```
$ history | more             # display previous commands a screen at a time
$ history | tail -10  # show the 10 most recently run commands
```

명령어 다시 실행하기

명령어를 다시 실행하려면, 일단 '위쪽' 화살표 키를 사용해 이전에 실행한 명령어를 탐색한다. 가장 최근 실행한 명령어가 먼저 표시된다. 또한 history 명령어의 출력에 표시된 숫자를 입력하고 !(느낌표)를 추가해도 된다.

```
$ !196
ls happy_quotes
happy_quotes
```

이전에 사용한 디렉터리로 이동

cd /user/bin 등 다른 디렉터리로 이동하거나 인수 없이 cd만 사용해 홈 디렉터리로 돌아가는 것은 쉽지만, 직전 디렉터리로 이동하려면 어떻게 해야 할까? 다음과 같이 하면 된다.

```
$ cd -
```

이 트릭을 사용하면 단순히 방금 있었던 위치로 돌아가고자 할 때 긴 경로 이름을 입력할 필요가 없다.