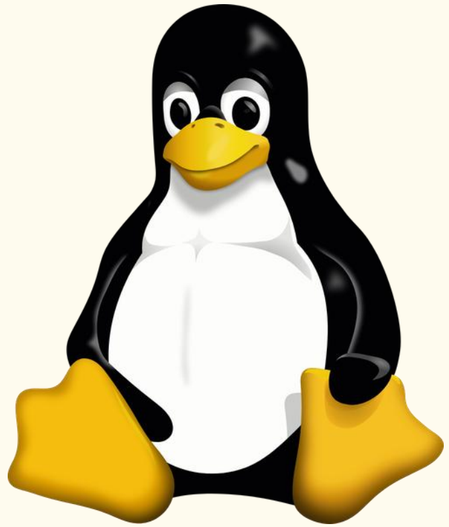
명령어에 대하여

리눅스

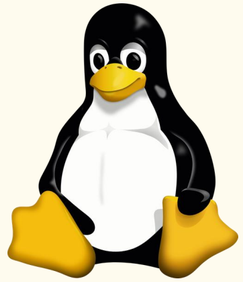
2026-09-18
박순민

Contents

리눅스?



UNIX 기반 운영체제



보안성


debian
fedora
gentoo linux
KNOPPIX
Mandriva
MEPIS
Novell
redhat
slackware
SUSE
ubuntu
xandros

다양한 배포판

[명령어]

단독 명령어 사용

[명령어][경로]

```
ls /
```

```
cat /etc/passwd
```

특정 경로에 명령어 사용

02 기본구조

[명령어][옵션][경로]

```
~# ls -l /root
```

umask

NEW

umask

기본 권한

666

777

umask

기본 권한

umask

666

777

03 명령어

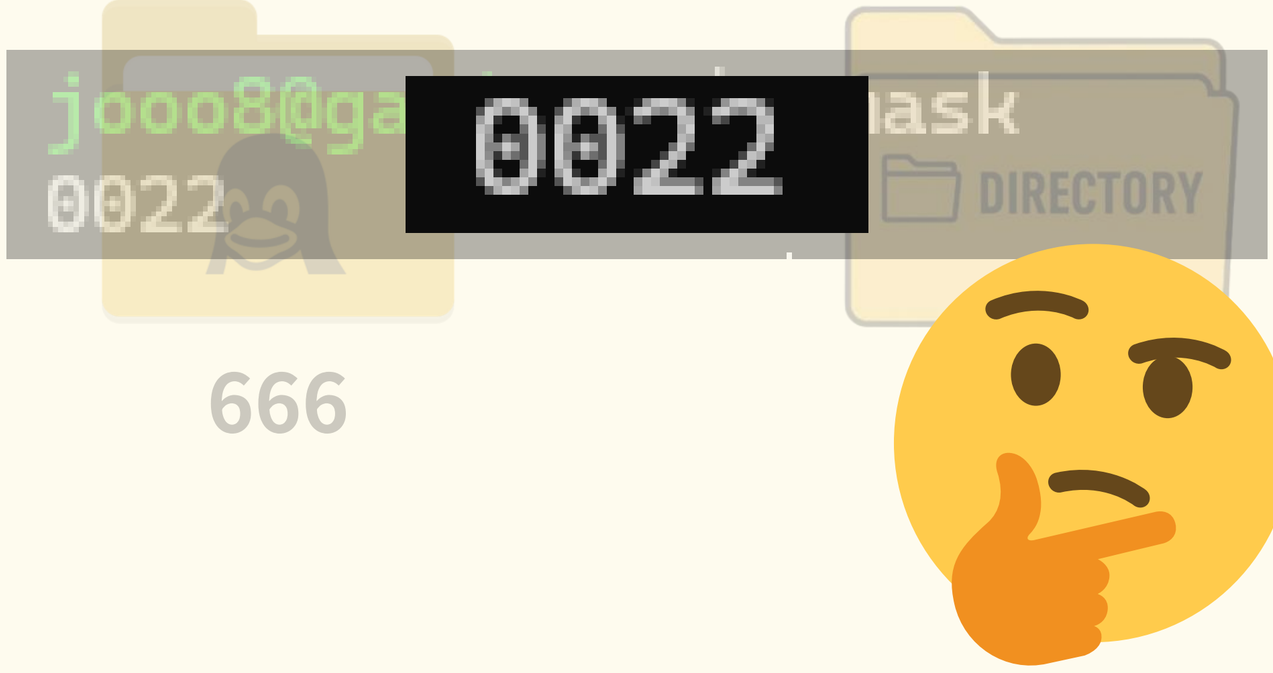
기본 권한

```
jooo8@gaming:~$ umask
0022
```

666

777

기본 권한



0022

1

0 0 2 2

특수 권한

setUid, setGid . . .

03 명령어

1

0 0 2 2

03 명령어

2

0 0 2 2

소유자 권한

r w x

03 명령어

3

0 0 2 2

그룹 권한

r w x

4

0 0 2 2

타인(Other) 권한

r w x

기본 권한 - umask

기본 권한 - umask

= NEW 기본 권한

기본 권한

666

777

umask

기본 권한

644

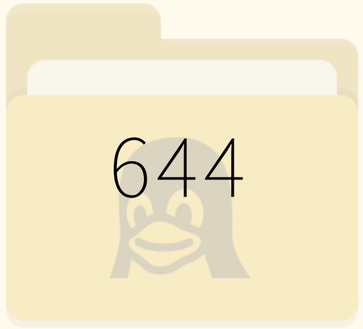
755

umask

NEW

umask

NEW 기본 권한



umask

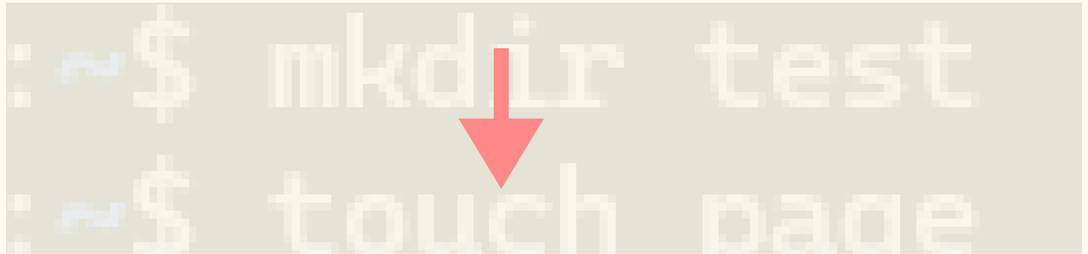
0022

```
:~$ mkdir test
:~$ touch page
```

umask

0022

```
drwxrwxrwx 2 jooo8 jooo8 4096 Sep 19 20:43 test
```




```
jooo8@gaming:~$ ls -ld test
drwxr-xr-x 2 jooo8 jooo8 4096 Sep 19 20:33 test
```

umask

0022

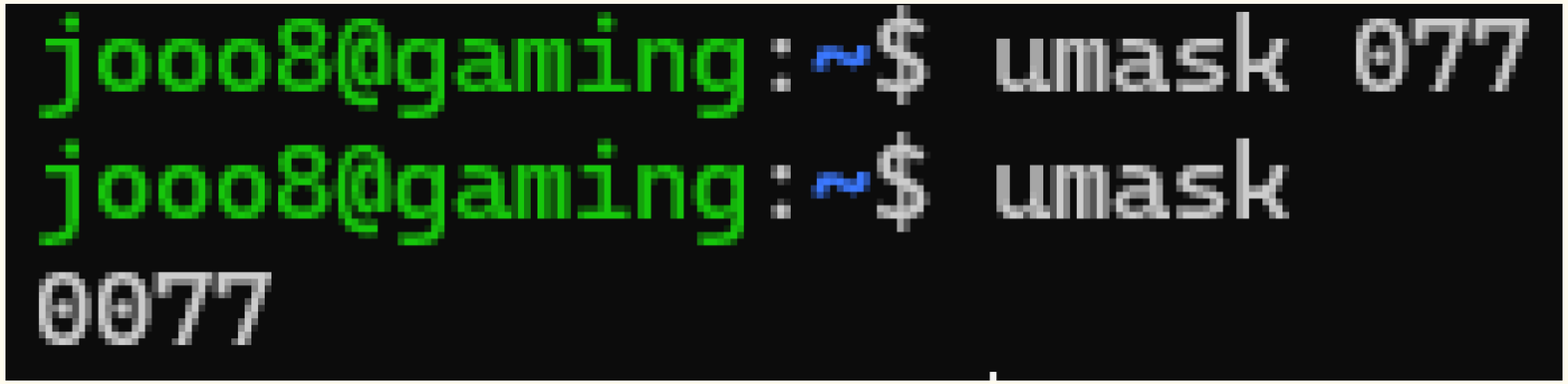
```
-rw-rw-rw- 1 jooo8 jooo8 0 Sep 19 20:43 page
```

```
:~$ mkdir test
:~$ touch page
```

```
jooo8@gaming:~$ ls -l page
-rw-r--r-- 1 jooo8 jooo8 0 Sep 19 20:33 page
```

umask

만약



0077

```
jooo8@gaming:~$ mkdir test1
jooo8@gaming:~$ touch page1
```

umask

```
0077
```

```
drwxrwxrwx 2 jooo8 jooo8 4096 Sep 19 20:43 test
```

```
drwx------ 2 jooo8 jooo8 4096 Sep 19 20:49 test1
```

700

umask

0077

```
-rw-rw-rw- 1 jooo8 jooo8 0 Sep 19 20:43 page
```

```
-rw------- 1 jooo8 jooo8 0 Sep 19 20:49 page1
```

600

umask

make

03 명령어

소스코드를 컴파일하고 프로그램을
빌드하기 위한 도구

make

매크로

소스코드를 컴파일하고 프로그램을 빌드하기 위한 도구

make

03 명령어

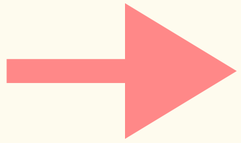
소스코드를 컴파일하고 프로그램을
빌드하기 위한 도구

make

03 명령어

도구

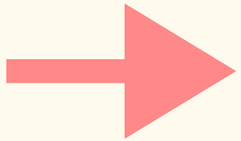
make



03 명령어

도구

make



사용 지침서

makefile

03 명령어

03 명령어

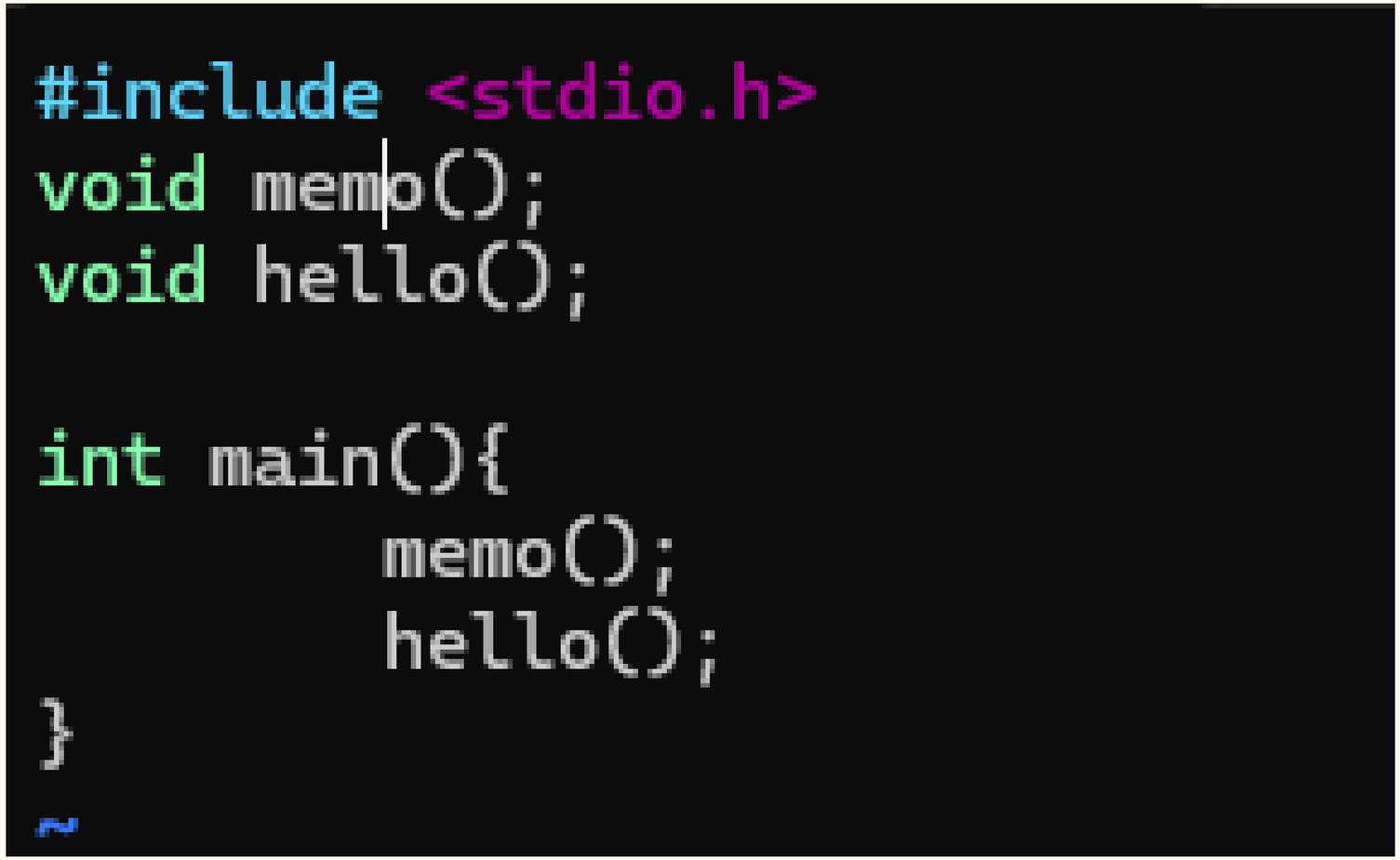
makefile

03 명령어

03 명령어

```
vi hello.c
vi memo.c
vi pass.c
```

03 명령어



03 명령어

```
/*** hello.c **/

#include <stdio.h>
void hello(){
    printf("Hello!\n");
}
~
```

03 명령어

```
#include <stdio.h>

void memo(){
        printf("Welcome to Null4u!\n");
}
~
```

03 명령어

```
gcc -c memo.c
gcc -c hello.c
gcc -c pass.c
```

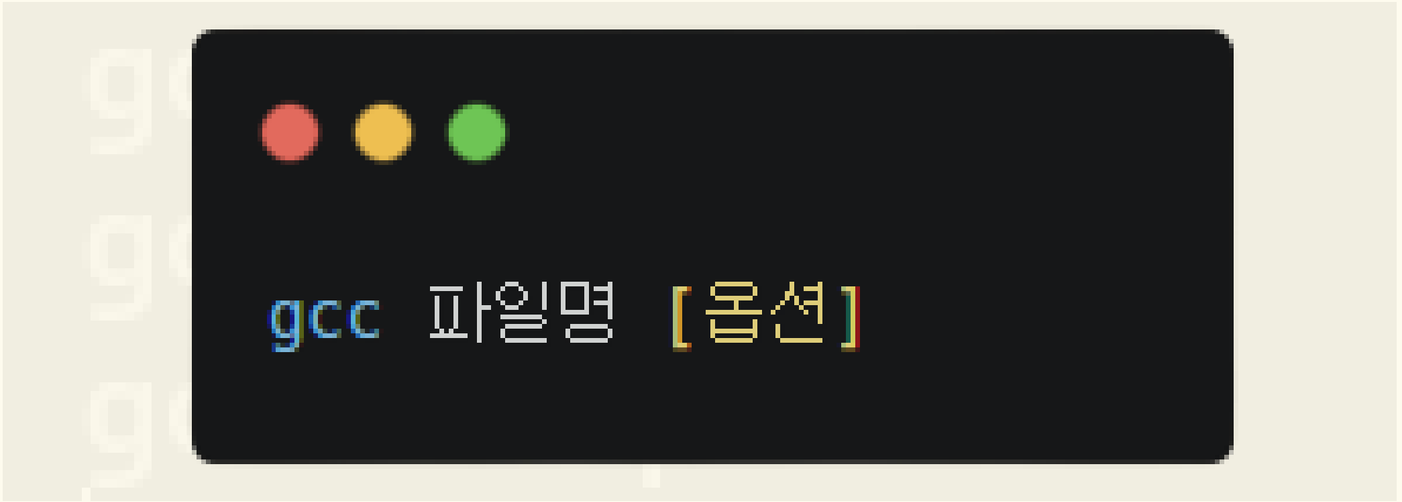
03 명령어

gcc?

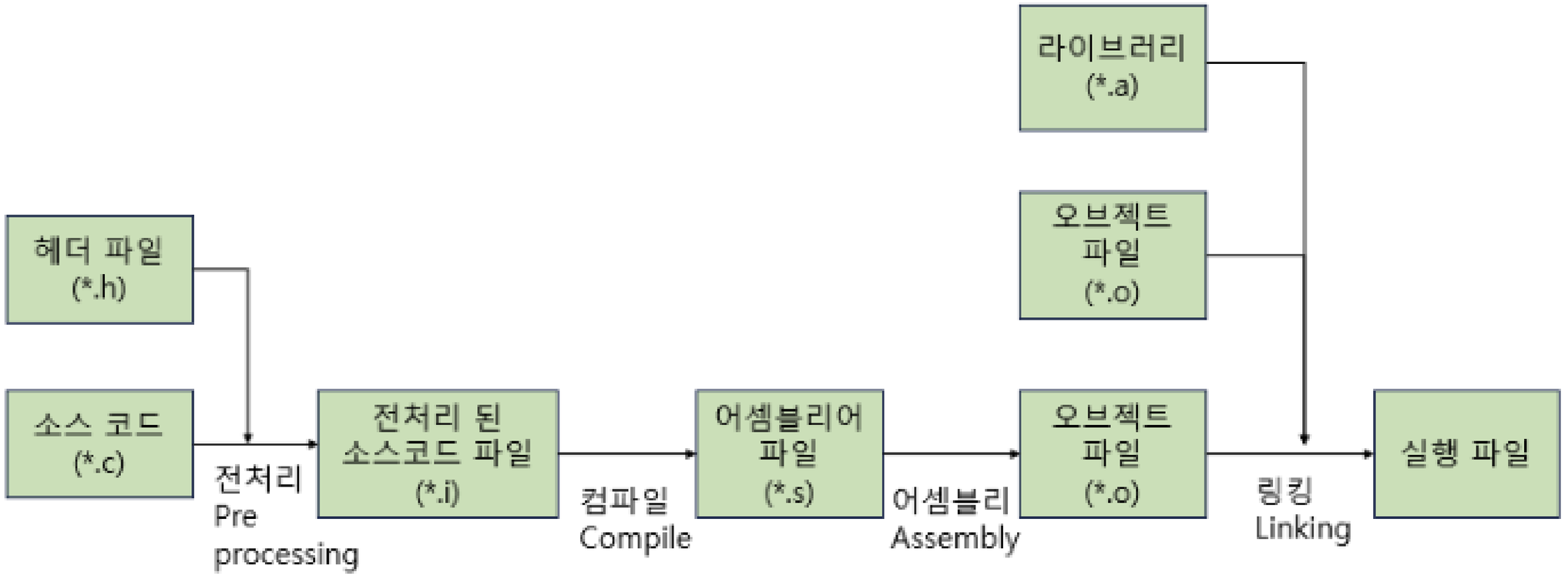
컴파일러

C, C++, Python 등등

03 명령어



03 명령어



[그림 1] 컴파일 과정

03 명령어

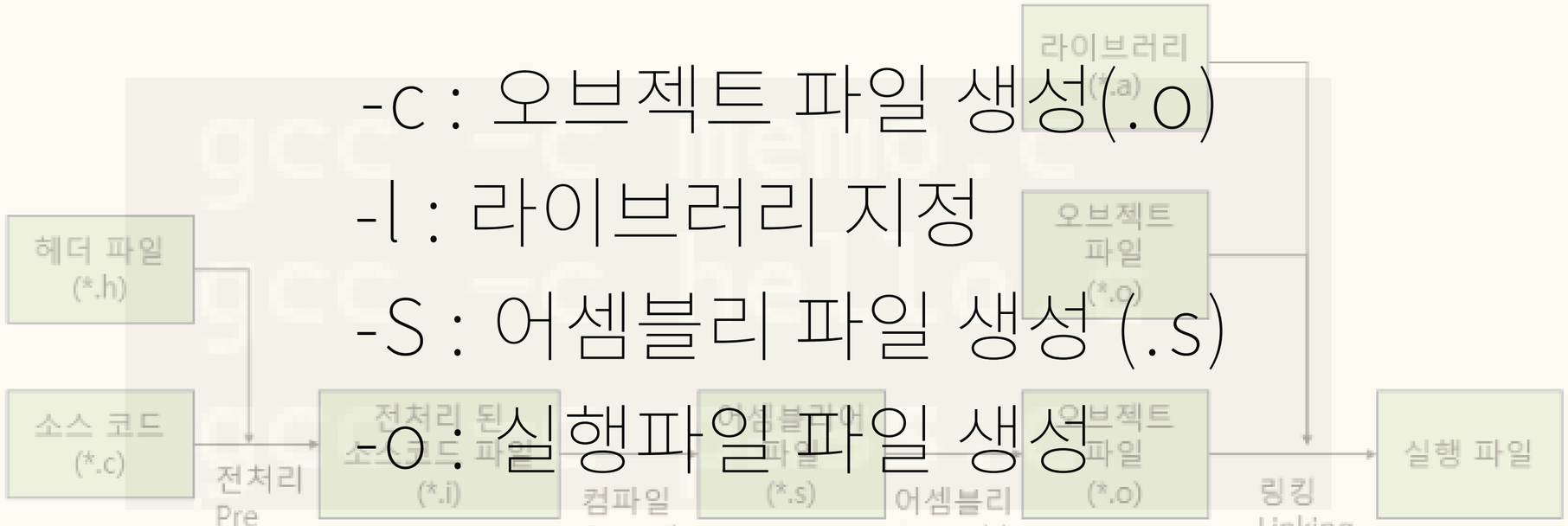
-c : 오브젝트 파일 생성(.o)

-l : 라이브러리 지정

-S : 어셈블리 파일 생성 (.s)

-O : 실행파일 파일 생성

-E : 전처리 단계 실행 (.i)



[그림 1] 컴파일 과정

03 명령어

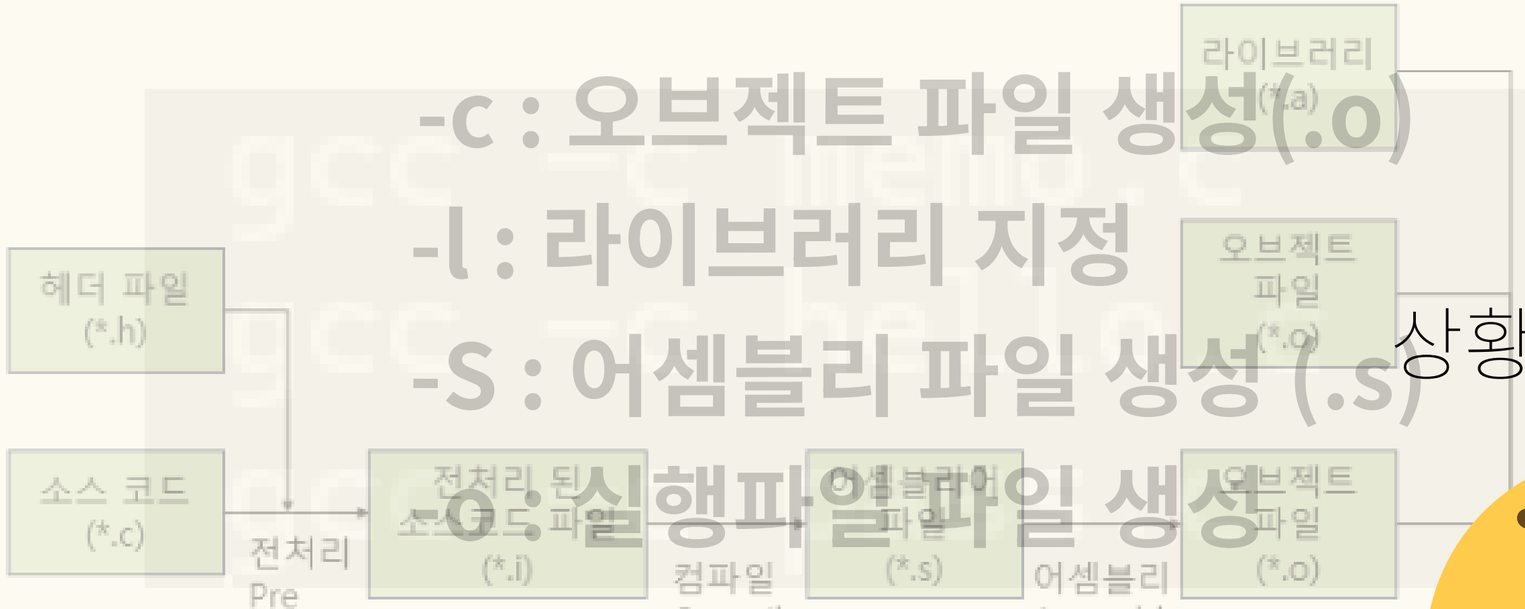
-c : 오브젝트 파일 생성(.o)

-l : 라이브러리 지정

-S : 어셈블리 파일 생성 (.s)

-o : 실행파일 파일 생성

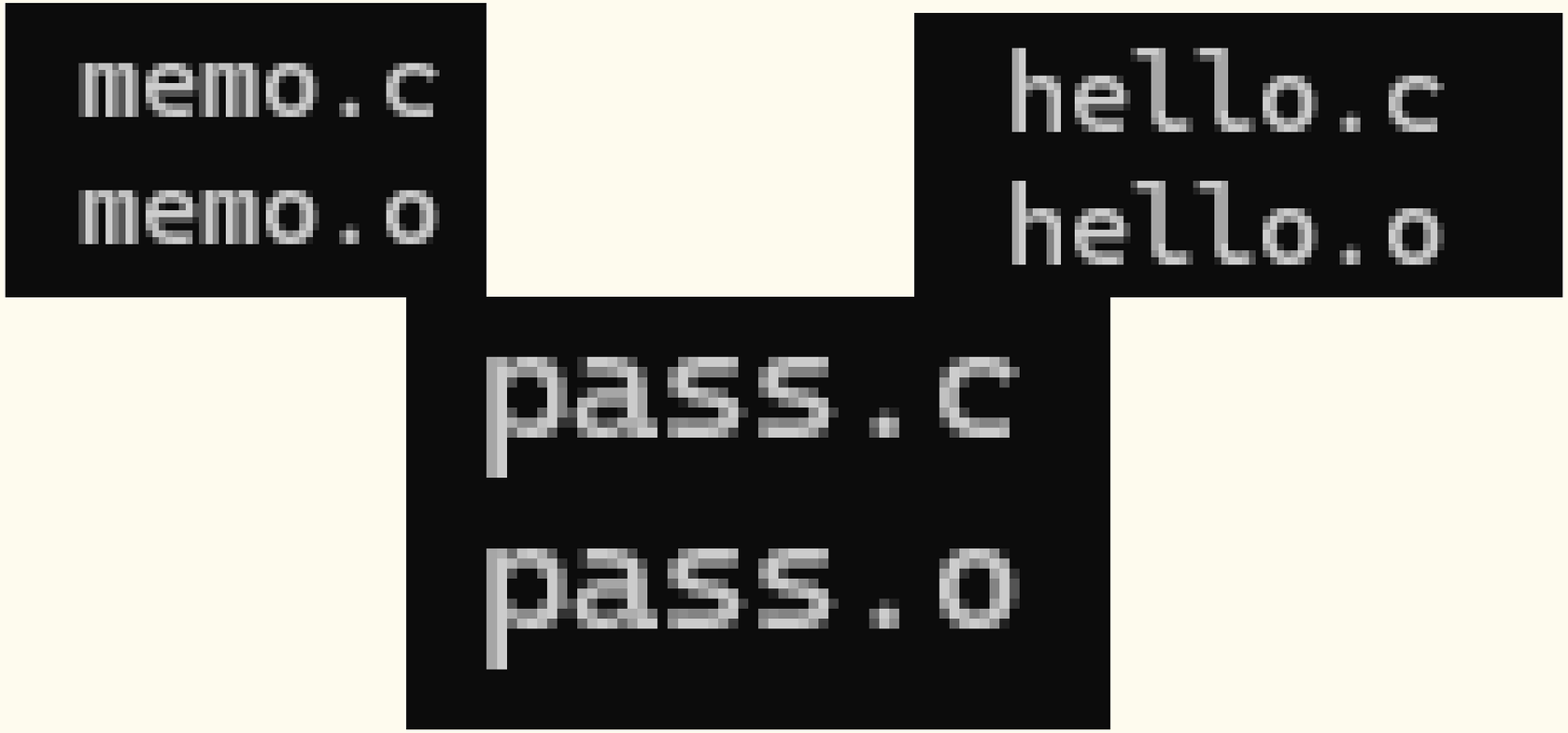
-E : 전처리 단계 실행 (.i)



[그림 1] 컴파일 과정

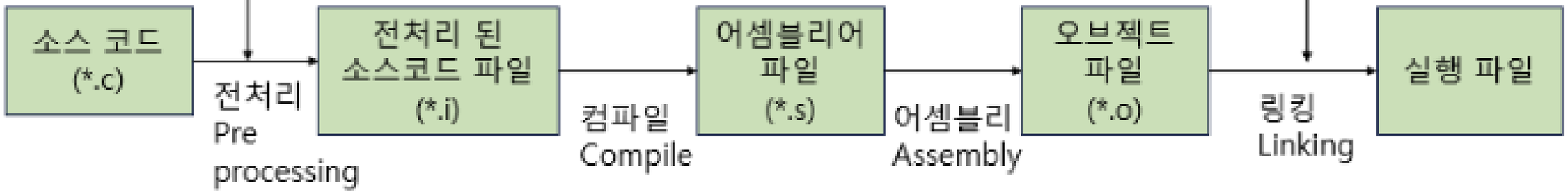
상황에 맞게...

03 명령어



03 명령어

왜 순서대로 안하지??



03 명령어

gcc 내부적으로
자동 처리

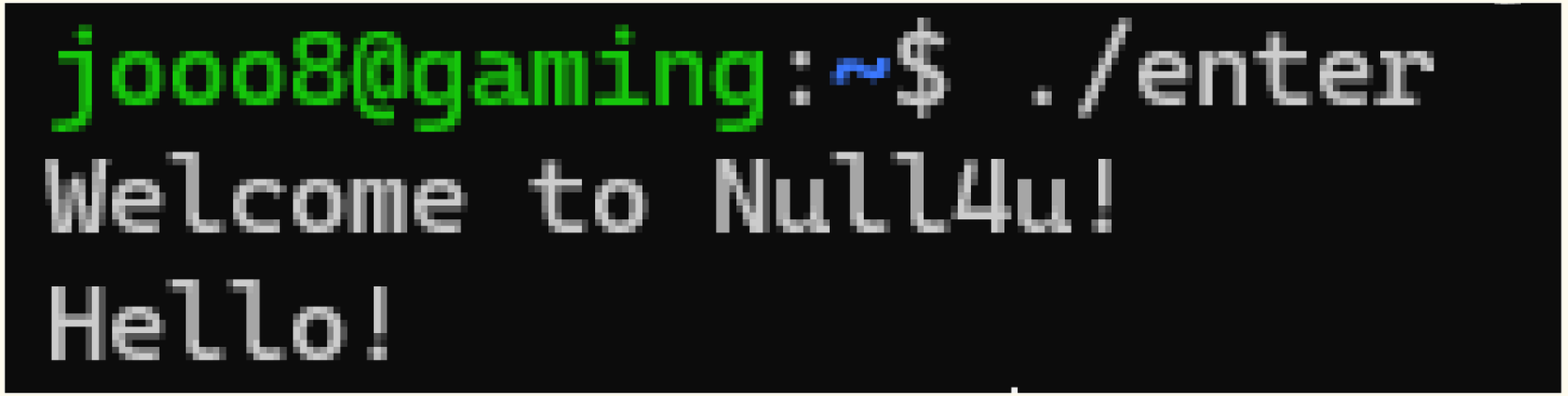
```
hello.c
  ↓ 전처리
(내부적으로 .i 생성)
  ↓ 컴파일
(내부적으로 .s 생성)
  ↓ 어셈블
hello.o
  ↓ 링킹
hello
```

03 명령어

실행 파일

```
gcc -o enter hello.o memo.o pass.o
```

실행



03 명령어

```
jooo8@gaming:~$ ./enter
Welcome to the Matrix!
Hello!
```

makefile도 가능

03 명령어

```
vi Makefile
```

```
enter2 : memo.o pass.o hello.o
	gcc -o enter2 memo.o pass.o hello.o
```

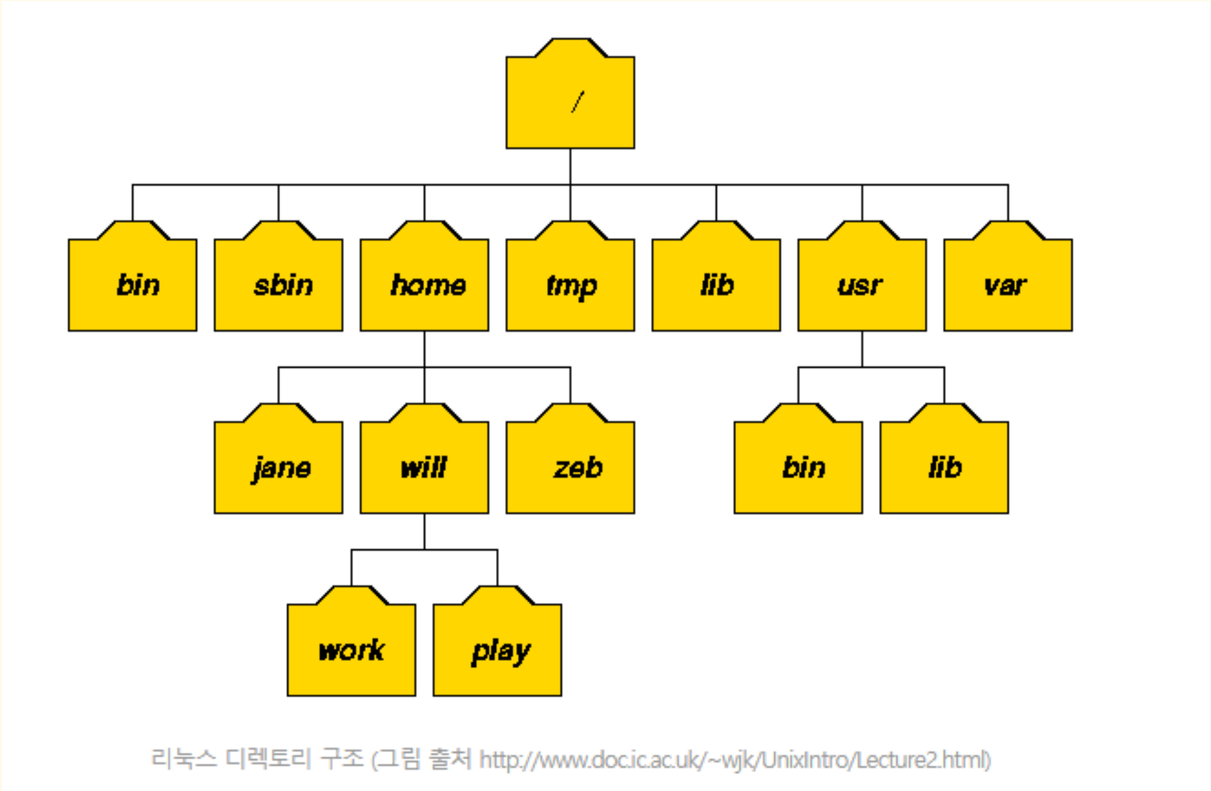
03 명령어

```
jooo8@gaming:~$ make
gcc -o enter2 memo.o pass.o hello.o
```

03 명령어

```
jooo8@gaming:~$ ./enter2
Welcome to Null4u!
Hello!
```

find



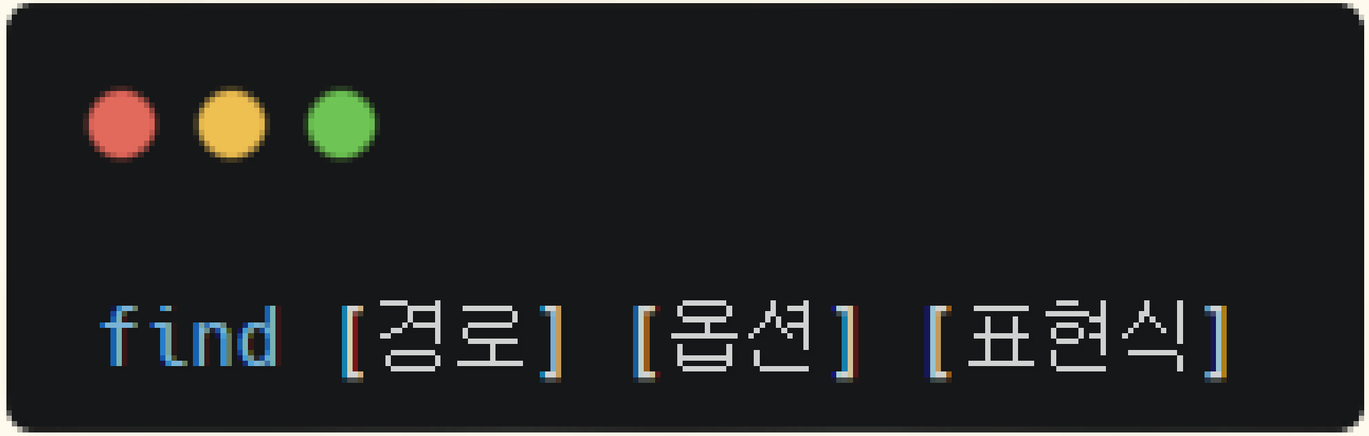
리눅스 디렉토리 구조 (그림 출처 http://www.doc.ic.ac.uk/~wjk/UnixIntro/Lecture2.html)

find

파일 시스템

03 명령어

find



find

[옵션]

-P	심벌릭 링크 자체 정보 사용
-L	심벌릭 링크에 연결된 파일 정보 사용
-H	심벌릭 링크를 따라가지 않으나 Command Lince Argument를 처리할 땐 예외
-D	디버그 메시지 출력

find

03 명령어

find

[표현식]

- name : 해당 이름의 파일을 찾음. 해당 이름에는 정규 표현식을 활용할 수 있음
- type : 지정된 파일 타입에 해당하는 파일 검색
- user : 해당 유저에게 속한 파일 검색
- empty : 빈 디렉토리 혹은 크기가 0인 파일 검색
- delete : 검색된 파일 혹은 디렉토리 삭제
- exec : 검색된 파일에 대해 지정된 명령 실행
- path : 지정된 문자열 패턴에 해당하는 경로에서 검색.
- print : 검색 결과를 출력. 검색 항목은 newline으로 구분. (기본 값)
- print0 : 검색 결과를 출력. 검색 항목은 null로 구분.
- size : 파일 크기를 사용하여 파일 검색.
- mindepth : 검색을 시작할 하위 디렉토리 최소 깊이 지정.
- maxdepth : 검색할 하위 디렉토리의 최대 깊이 지정.
- atime : n일 이내에 액세스된 파일을 찾음.
- ctime : n일 이내에 만들어진 파일을 찾음.
- mtime : n일 이내에 수정된 파일을 찾음.
- cnewer file : 해당 파일보다 최근에 수정된 파일을 찾음.

find

[표현식]

- name : 해당 이름의 파일을 찾음. 해당 이름에는 정규 표현식을 활용할 수 있음
- type : 지정된 파일 타입에 해당하는 파일 검색
- user : 해당 유저에게 속한 파일 검색
- empty : 빈 디렉토리 혹은 크기가 0인 파일 검색
- delete : 검색된 파일 혹은 디렉토리 삭제
- exec : 검색된 파일에 대해 지정된 명령 실행
- path : 지정된 문자열 패턴에 해당하는 경로에서 검색.
- print : 검색 결과를 출력. 검색 항목은 newline으로 구분. (기본 값)
- print0 : 검색 결과를 출력. 검색 항목은 null로 구분.
- size : 파일 크기를 사용하여 파일 검색.
- mindepth : 검색을 시작할 하위 디렉토리 최소 깊이 지정.
- maxdepth : 검색할 하위 디렉토리의 최대 깊이 지정.
- atime : n일 이내에 액세스된 파일을 찾음.
- ctime : n일 이내에 만들어진 파일을 찾음.
- mtime : n일 이내에 수정된 파일을 찾음.
- cnewer file : 해당 파일보다 최근에 수정된 파일을 찾음.

-name과 -type을 소개

find -name

```
jooo8@gaming:~$ find -name '*.txt'
./test.txt
./tast3.txt
./test3.txt
./test2.txt
./test1.txt
./tust.txt
./tast.txt
./tast2.txt
./tast1.txt
```

find -name

```
jooo8@gaming:/home$ find -name '*.sh'
./jooo8/for_sum.sh
./jooo8/while_sum.sh
./jooo8/cmd_par.sh
./jooo8/case_esac.sh
./jooo8/until_sum.sh
./jooo8/if_else2.sh
./jooo8/if_else3.sh
./jooo8/fun.sh
./jooo8/add_mul.sh
./jooo8/break.sh
./jooo8/hello.sh
./jooo8/home.sh
./jooo8/in_out.sh
./jooo8/if_else1.sh
./jooo8/seq_sum.sh
./jooo8/sum_mul.sh
./jooo8/user_name.sh
```

find -name

```
jooo8@gaming:~$ find -name 't[ae]st.*'
./test.txt
./tast.txt
```

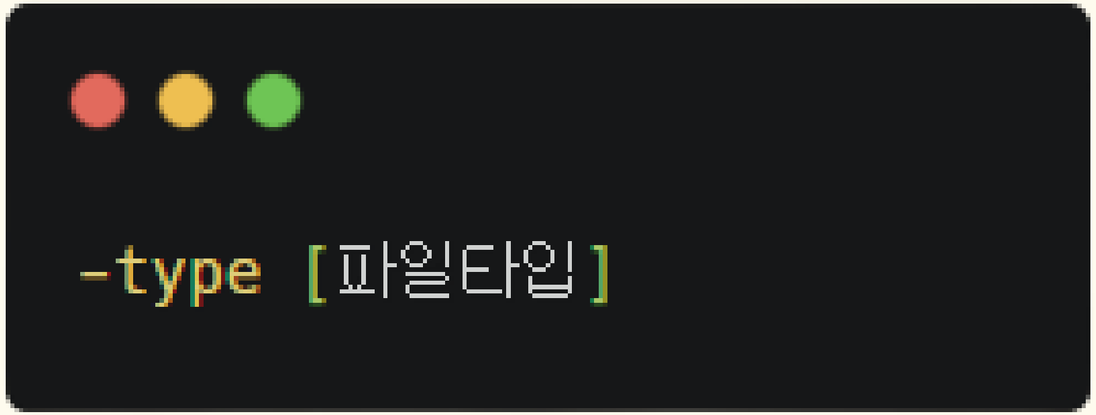
```
jooo8@gaming:~$ find -name 'enter4' -ok rm {} \;
< rm ... ./enter4 > ? y
```

find -name

```
jooo8@gaming:~$ find -name 'hello.c' -exec cat {} \;
/*** hello.c **/

#include <stdio.h>
void hello(){
    printf("Hello!\n");
}
```

find -name

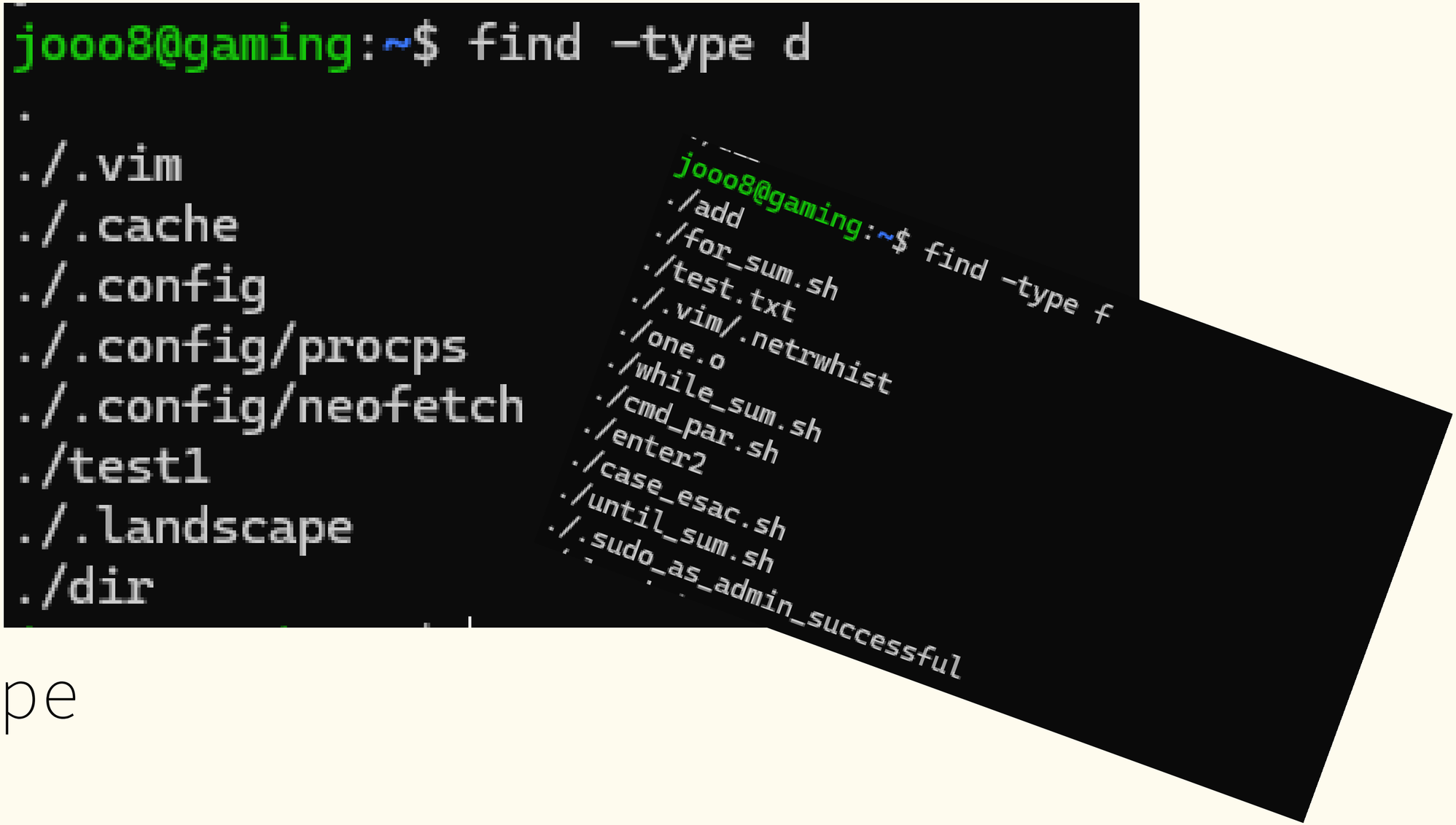


find -type

[파일타입]

d : 디렉토리
f : 일반 파일
l : 심벌릭 링크
etc....

find -type


jooo8@gaming:~$ find -type d
.
./.vim
./.cache
./.config
./.config/procps
./.config/neofetch
./test1
./.landscape
./dir
jooo8@gaming:~$ find -type f
./add
./for_sum.sh
./test.txt
./.vim/.netrwhist
./one.o
./while_sum.sh
./cmd_par.sh
./enter2
./case_esac.sh
./until_sum.sh
./.sudo_as_admin_successful

find -type

awk

03 명령어

hello	1234	old
my	name	pass
is	baby	SQL
soon	min	Java

레코드로 읽고,

awk

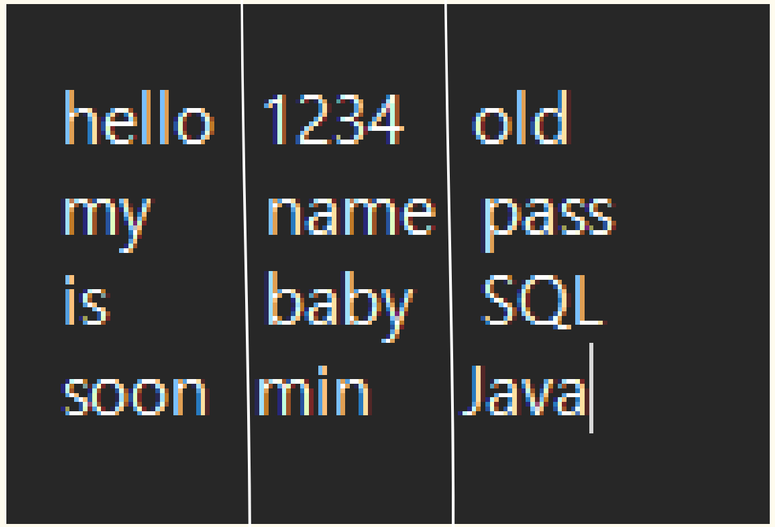
03 명령어

hello	1234	old
my	name	pass
is	baby	SQL
soon	min	Java

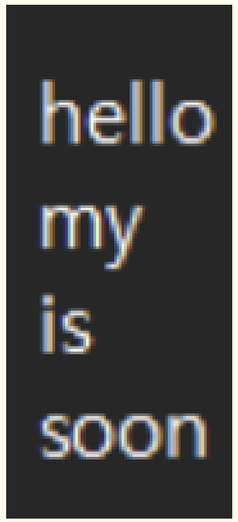
여러 개의 열로 나눔

awk

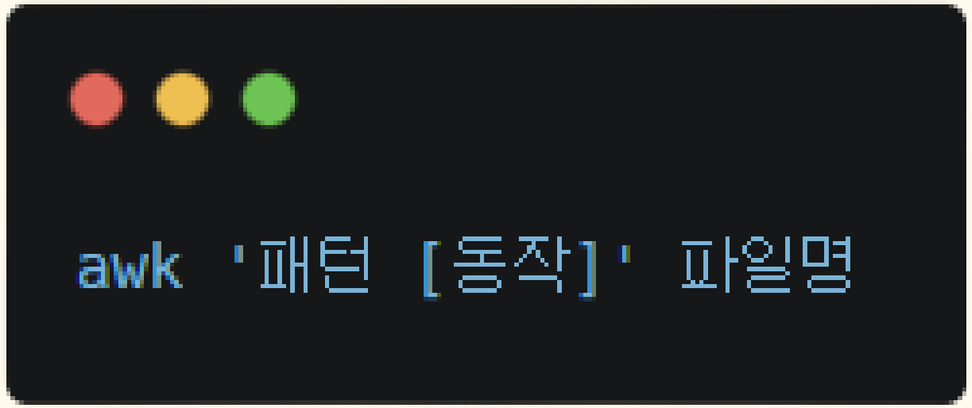
03 명령어


hello 1234 old
my name pass
is baby SQL
soon min Java

조건


hello
my
is
soon

awk



awk

03 명령어

[동작]

내장 변수

$0 : 파일 전체 출력

$1 ~ .. : 각 필드

etc....

[패턴]

조건식, 연산식

특정 키워드

NR : 읽은 행 번호

etc....

awk

03 명령어

```
//test.txt

1 2 5 6 7
3 4 8 9 10
hello my name is soonmin
Welcome to semyung computer high school
~
```

awk

```
jooo8@gaming:~$ awk '{print $5}' test.txt


7
10
soonmin
high
```

awk

```
jooo8@gaming:~$ awk '{print $5}' test.txt


7
10
soonmin
high
```

```
jooo8@gaming:~$ awk 'NR == 5 {print $0}' test.txt
hello my name is soonmin
```

awk

```
jooo8@gaming:~$ awk '{print $5}' test.txt


7
10
soonmin
high
```

```
jooo8@gaming:~$ awk 'NR == 3 {print $2 * $4}' test.txt
12
```

```
jooo8@gaming:~$ awk 'NR == 5 {print $0}' test.txt
hello my name is soonmin
```

awk

sed

03 명령어

```
name        phone
Pak         010 - 1111 - 1111
kim         010 - 2222 - 2222
Lee         010 - 3333 - 3333
Yoo         010 - 4444 - 44444
~
```

sed

03 명령어

명령어

```
name      phone
Pak       010 - 1111 - 1111
kim       010 - 2222 - 2222
Lee       010 - 3333 - 3333
Yoo       010 - 4444 - 4444
~
```

sed

03 명령어

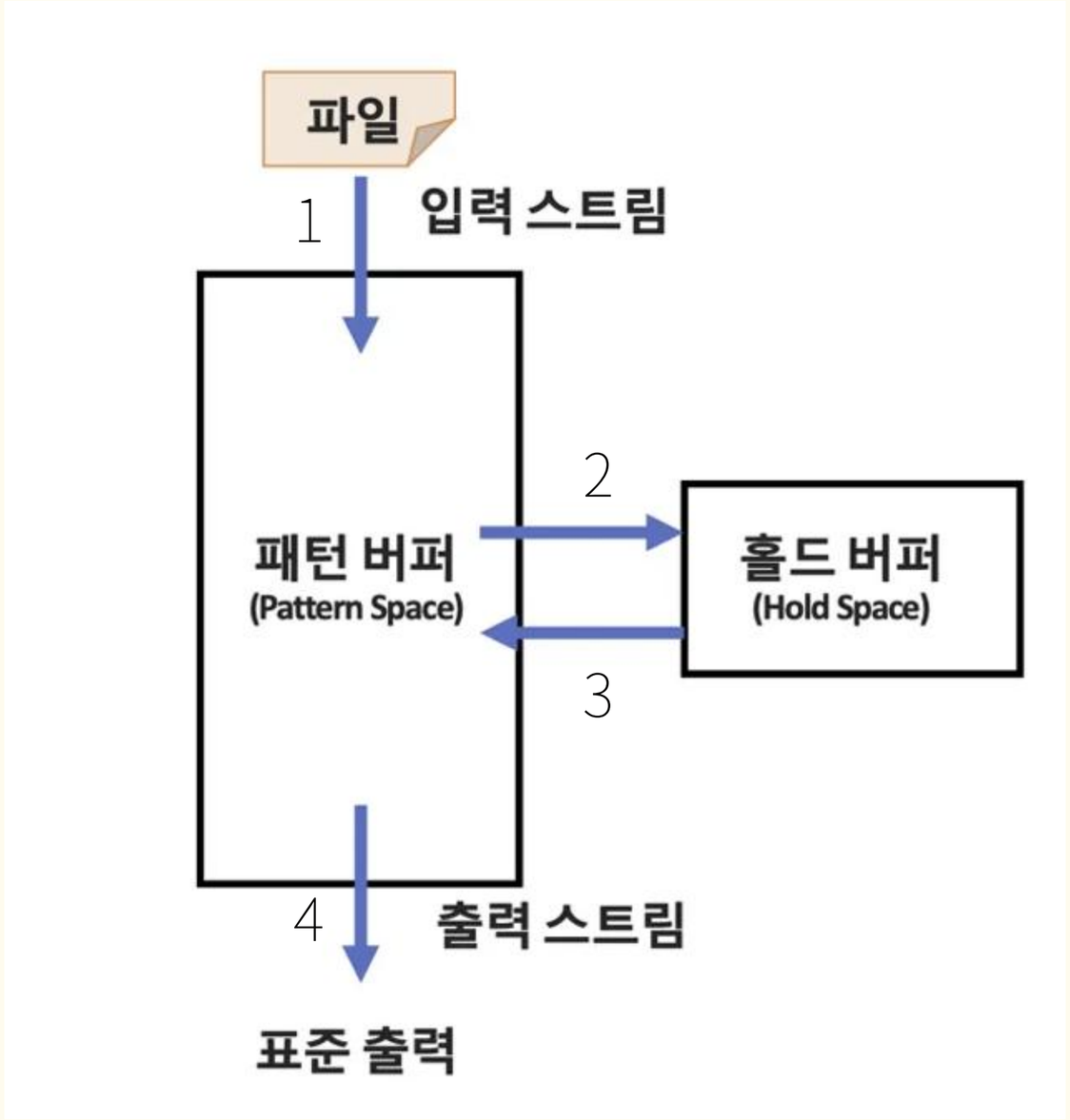
```
name      phone
Pak       010 - 1111 - 1111
kim       010 - 2222 - 2222
Lee       010 - 3333 - 3333
Yoo       010 - 4444 - 4444
~
```

치환 

검색 

삭제 

sed



sed 동작원리

sed

```
name      phone
Pak       010 - 1111 - 1111
kim       010 - 2222 - 2222
Lee       010 - 3333 - 3333
Yoo       010 - 4444 - 4444
~
```

sample 파일

sed

03 명령어

```
sed [옵션] '명령어' [파일명]
```

sed

패턴 버퍼의 내용 출력 X

```
jooo8@gaming:~$ sed -n '1p' sample
name      phone
```

출력

패턴 버퍼의 내용 출력 X

↓

만약 허용한다면?

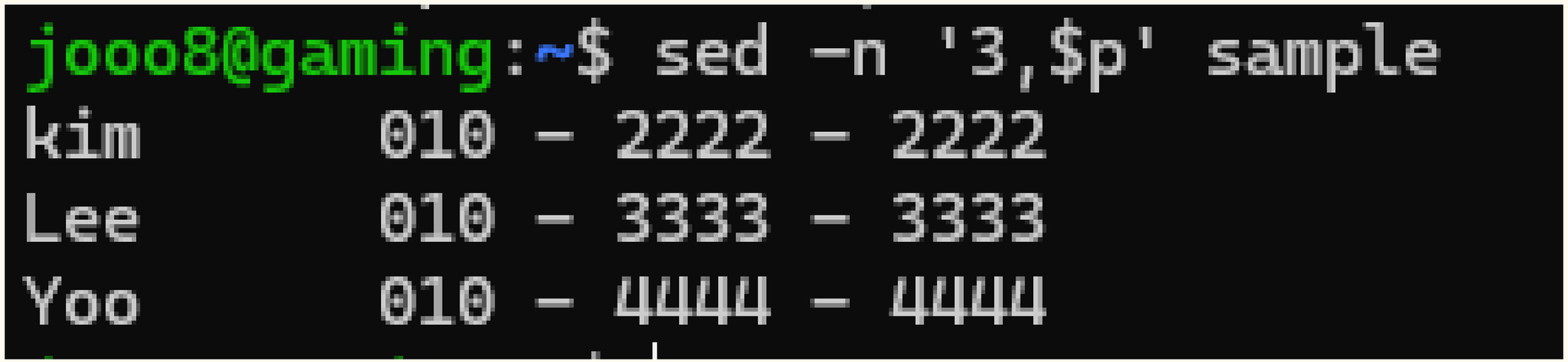
출력

```
jooo8@gaming:~$ sed '1,$p' sample
name      phone
name      phone
Pak       010 - 1111 - 1111
Pak       010 - 1111 - 1111
kim       010 - 2222 - 2222
kim       010 - 2222 - 2222
Lee       010 - 3333 - 3333
Lee       010 - 3333 - 3333
Yoo       010 - 4444 - 4444
Yoo       010 - 4444 - 4444
```

패턴 버퍼의 내용까지 같이 출력

출력

끝까지


jooo8@gaming:~$ sed -n '3,$p' sample
kim 010 - 2222 - 2222
Lee 010 - 3333 - 3333
Yoo 010 - 4444 - 4444

출력

자동 출력 끄기

```
jooo8@gaming:~$ sed -n '/Lee/p' sample
Lee         010 - 3333 - 3333
```

출력

```
sed 's/원본문자열/바꿀문자열/g' 파일명
```

치환

```
jooo8@gaming:~$ sed 's/2222/1975/g' sample
name      phone
Pak       010 - 1111 - 1111
kim       010 - 1975 - 1975
Lee       010 - 3333 - 3333
Yoo       010 - 4444 - 4444
```

```
jooo8@gaming:~$ sed -n 's/1111/1010/gp' sample
Pak       010 - 1010 - 1010
```

치환

```
sed '/삭제 할 내용/d' 파일명.txt
```

삭제

```
jooo8@gaming:~$ sed '3d' sample
name        phone
Pak         010 - 1111 - 1111
Lee         010 - 3333 - 3333
Yoo         010 - 4444 - 4444
```

```
jooo8@gaming:~$ sed /Yoo/d sample
name        phone
Pak         010 - 1111 - 1111
kim         010 - 2222 - 2222
Lee         010 - 3333 - 3333
```

삭제

04 느낀점

명령어를 외우는 것은 힘든 일이고

미친짓이구나

QnA

THANK YOU

들어주셔서 감사합니다.