EDDI
Electronic Design
Development Institute

에디로봇아카데미

임베디드 마스터 Lv1 과정

제 4기

2022. 08. 26

한정훈

CONTENTS

EDDI
Electronic Design
Development Institute

Git 기본설정

1) Git 기본 설정

>git commit -am = “메세지”(여기까지는 로컬영역 - 내 PC에만 저장)
>git push origin main = 원격 저장소에 업로드
>git pull origin main = 원격 저장소의 내용 업데이트(가져오기,당겨오기)

git 기본 설정

```
git status: 현재상태
    추적되지 않는 정보: 빨간색 텍스트

git add 파일명 = 추가하기(?) 추적을 시작해라(?)
git commit -am "[한정훈] OT 테스트 커밋
git push origin main
❯ git push origin main
Username for 'https://github.com': hunyhan
Password for 'https://hunyhan@github.com':

깃 업데이트 방법 (master내용 가져오기, main 업데이트 이후가능)
❯ git pull origin main
```

2) terminal 단축어

EDDI
Electronic Design Development Institute

1) 컴파일 방법

>예를들어 test.c 를 컴파일 하면 => gcc test.c => a.out
  결과물 실행파일 명을 바꾸고 싶으면,
=> gcc -o 파일명 test.c => 파일명 실행파일이 나온다.

2) 파일 지우기/폴더 지우기

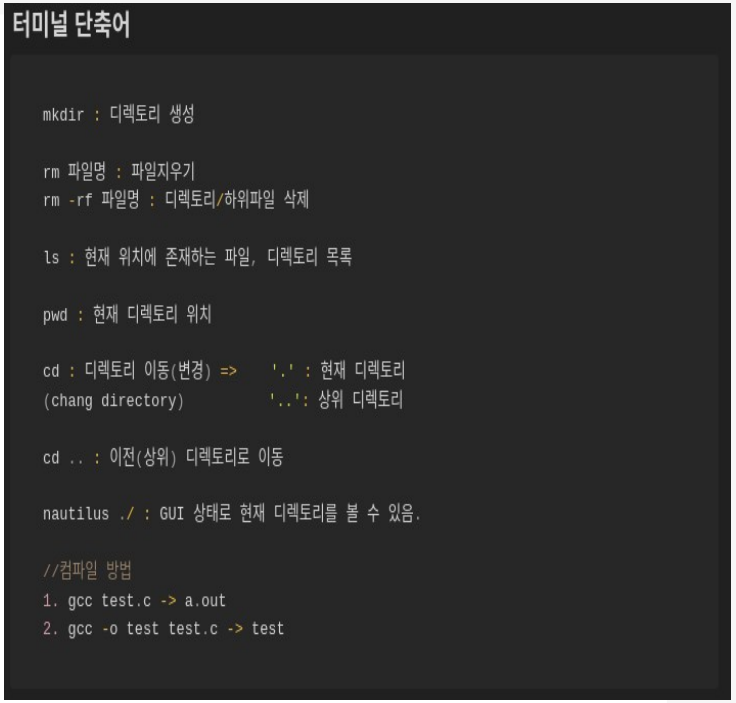
> 파일을 지울 때는 (rm 파일명)
> 폴더와 하위파일을 지울 때는 (rm -rf 폴더명)

3) 터미널에서 외부의 복사 내용 반영하기

> 오른쪽 마우스로 클릭 후 복사 붙여넣기가 가능.
> Ctrl + Shift + C / Ctrl + Shift + V
(이유는 터미널에서는 Ctrl + C는“명령중지”기능이 별도로 있기 때문.)
>이전에 무한루프가 돌 경우에 Ctrl + C를 입력하니 멈추고 사용 가능했음.

*)추가

>혹시나 Ctrl + S를 하는경우 터미널이 멈추므로 터미널 재시작 하기.



2) terminal 단축어

4) clear => Ctrl + L

터미널 단축어

```
mkdir : 디렉토리 생성

rm 파일명 : 파일지우기
rm -rf 파일명 : 디렉토리/하위파일 삭제

ls : 현재 위치에 존재하는 파일, 디렉토리 목록

pwd : 현재 디렉토리 위치

cd : 디렉토리 이동(변경) =>    '.' : 현재 디렉토리
(chang directory)          '..': 상위 디렉토리

cd .. : 이전(상위) 디렉토리로 이동

nautilus ./ : GUI 상태로 현재 디렉토리를 볼 수 있음.

//컴파일 방법
1. gcc test.c -> a.out
2. gcc -o test test.c -> test
```

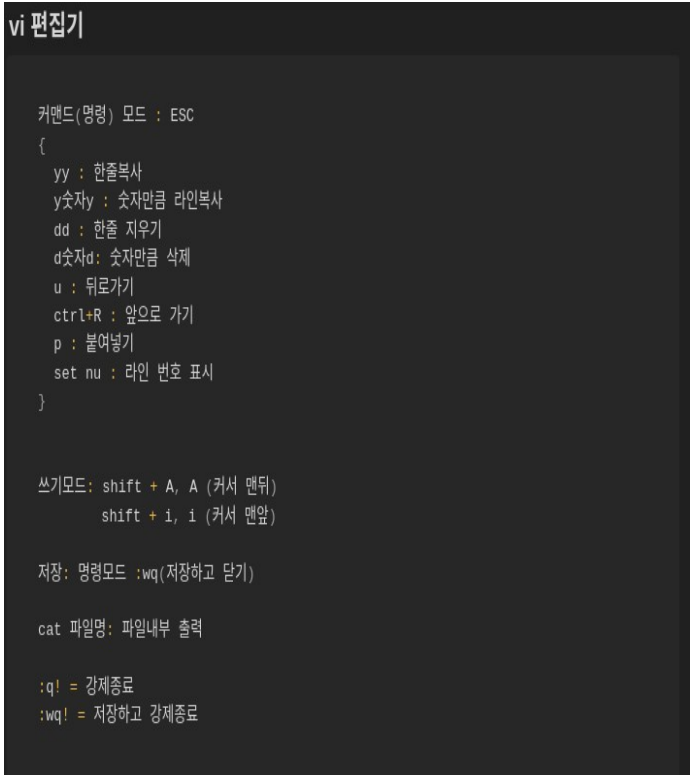
3) vi 편집기 단축어

1) 커맨드 모드 : ESC

>커맨드 모드에서는 복사, 붙여넣기, 지우기, 뒤로가기 등 간편한 단축키로 사용 가능.
다만, “라인 번호 표시”의 경우는 :set nu으로 아래 텍스트로 작성하여 실행한다.

2) 터미널vi 편집기에서 외부의 복사 내용 반영하기

> 오른쪽 마우스로 클릭 후 복사 붙여넣기가 가능.
> Ctrl + Shift + C / Ctrl + Shift + V
= 터미널과 같음.



4) 변수

```c
include <stdio.h>

// void main(void)를 적는 경우가 있는데
// 펌웨어 코드와 다르게 애플리케이션 코드를 만드는 경우에는
// 이 부분에 void 작성하면 안됩니다.
// 이유가 뭘까요 ? 리턴값이 운영체제로 가기 때문에
// 실제 이 프로그램을 실행해서 관리하는 부모 프로세스가
// 해당 프로세스가 정상 종료되었는지
// 비정상 종료되었는지 파악하기 위한 용도로 리턴값을 사용합니다.
```

*엔트리 포인트란?
제어가 운영체제에서 => 프로그램 으로 가는 것 이다.
프로세서가 프로그램이나 코드에 진입해서 실행을 시작한다.

궁금한점 = 엔트리 포인트의 동작 방식에 대해 정확하게 이해가 가지 않습니다! 프로그램 동작 시점을 말하는 걸까요?

프로그램 : 실행파일
프로세스 : 실행파일 동작중인 상태
프로세서: 프로그램을 동작시켜주는 하드웨어

1)void main / int main / main (운영체제가 있는 애플리케이션 코드의 경우)

>보충내용
셋은 차이가 없다. 하지만 내부적으로 바라볼때는 다르다.
함수(main) 앞에 붙는 데이터형(int, void)은 리턴값의 데이터 형이다.
-int main() 하면 main()함수가 종료할때 정수형 값을 리턴하겠다는 뜻이다.
-void main() 하면 main()함수가 종료할때 아무 값도 리턴하지 않겠다는 뜻이다.
-main() == void mian()
main() 함수는 프로그램의 *엔트리 포인트이며 운영체제가 실행시켜주는 함수이다.
따라서 main()함수의 리턴값은 운영체제가 받는다. 선생님께서 말씀해주신 내용이 이 내용인 것 같다.
“운영체제는 프로그램이 종료할때 main()함수의 리턴값을 받아보고 왜 종료되었는가 판단.
보통의 경우 0을 리턴하면 프로그램이 정상적으로 실행을 마치고 종료한것으로 간주한다.”

4) 변수

EDDI
Electronic Design Development Institute

```
include <stdio.h>

// void main(void)를 적는 경우가 있는데
// 펌웨어 코드와 다르게 애플리케이션 코드를 만드는 경우에는
// 이 부분에 void 작성하면 안됩니다.
// 이유가 뭘까요 ? 리턴값이 운영체제로 가기 때문에
// 실제 이 프로그램을 실행해서 관리하는 부모 프로세스가
// 해당 프로세스가 정상 종료되었는지
// 비정상 종료되었는지 파악하기 위한 용도로 리턴값을 사용합니다.
```

```
    // 이런 부분에 숫자를 적으면
    // 해당 숫자가 무엇을 하는지 파악하기 위해 상당히 많은 비용을 소모하게 될 수도 있음.
    // 그러므로 맨 위의 #define 혹은 enum등을 활용해서
    // 이와 같은 처리를 사람이 알아볼 수 있게 만들어주는 것이 중요함.
 return SUCCESS;
```

“운영체제는 프로그램이 종료할때 main()함수의 리턴값을 받아보고 왜 종료되었는가 판단.
보통의 경우 0을 리턴하면 프로그램이 정상적으로 실행을 마치고 종료한것으로 간주한다.”

따라서 아래와 같이 사람이 알아볼 수 있게 #define 혹은 enum등을 활용해서 처리한다.

4) 변수

EDDI
Electronic Design Development Institute

2) 변수란?

- 특정한 데이터타입을 저장할 수 있는 메모리 (DRAM, 상황에 따라 SRAM이 될 수 있음) 공간
- 실제로 int num = 3; 이라는 코드는 어셈으로 mov(배치한다?) 명령어로 바뀌면서
  Num에 해당하는 메모리 공간에 배치(저장)한다라는 개념.

3) printf()는 괄호 내부의 정보를 출력한다.
-괄호 내부에 %d(정수), %s(문자열), %c(문자1개), %f(소수 float타입), %lf(double타입), %u(long타입)을 처리한다.

ex) printf(“num = %d\n”, num); => 출력 = “num = 3”