커맨드 라인 인자

2020년 11월 27일 금요일 17:48

커맨드 라인 인자

- 커맨드 라인에서 프로그램 실행할 때 인자들을 넣어주는 방법
- 예

```
> filecopy.exe a.txt b.txt
```

프로그램 이름 / 커맨드 라인 인자 (두 개)

- 저렇게 들어온 인자들 main( )함수의 매개변수에서 읽어올 수 있음

• 커맨드 라인 인자를 받는 main( ) 함수

```
int main(int argc, const char* argv[])
{
}
```

- argc는 들어온 인자의 수
  - 이 수에는 실행한 파일의 이름까지 포함

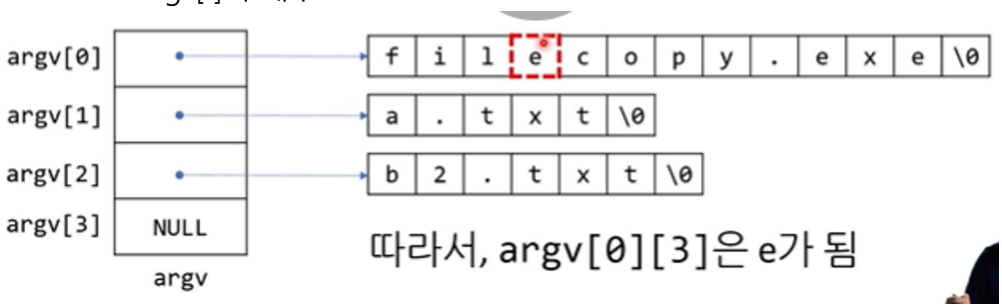
```
> filecopy.exe a.txt b2.txt
```

  - 위의 경우 argc는 3임
    - 첫 번째 인자: 실행한 파일의 경로
    - 나머지 인자: 그 후에 따라온 2개의 인자
- argv는 char 포인터 배열
  - argv[argc + 1]로 생성됨
  - argv[0]: 첫 번째 요소에는 실행 파일의 이름이 들어감
  - argv[1] ~ argv[argc-1]: 커맨드 라인 인자들이 순차적으로 들어옴
  - argv[argc]: NULL이 들어감
    - 널이 들어가 있기 때문에 argv 관련 반복문을 만들 때 argc가 아닌 null을 사용해도 된다

```
> filecopy.exe a.txt b2.txt
```

argv[0] argv[1] argv[2]

- const char* argv[ ]는 포인터의 배열
  - 각 포인터는 그냥 C 스타일 문자열
    1. 커맨드 라인에 들어온 값을 프로그램 실행할 때 만든 프로세스의 메모리 어딘가에 저장하고
       - 프로세스의 메모리: 해당 프로그램이 소유하고 있는 메모리
    2. 그 주소들을 모아 argv[ ] 배열에 넣어 보내주는 것
  - const char* argv[ ]의 내부



        - 위의 예의 경우 이렇게 들어감
- 커맨드 라인 인자의 메모리 뷰
    - 커맨드 라인에 있는 모든 것을 그대로 메모리에 복사 후에 각 인자가 시작하는 곳의 주소들을 넘겨준 것일까?
        - 이러러면 공백 문자가 메모리에 남아있어야 한다

```
> filecopy.exe a.txt        b2.txt
```

공백 8개

f	i	l	e	c	o	p
y	.	e	x	e	\0	a
.	t	x	t	\0		
					b	2
.	t	x	t	\0		

    - **<u>실제 메모리에는 공백문자가 저장되지 않는다</u>**

```
> filecopy.exe a.txt        b2.txt
```

f	i	l	e	c	o	p
y	.	e	x	e	\0	a
.	t	x	t	\0	b	2
.	t	x	t	\0		

- 입출력 리디렉션과 다르다
    - 입출력 리디렉션은 커맨드 라인 인자로 안 들어옴

```c
int main(int argc, const char* argv[])
{
    int i;

    printf("argc: %d\n", argc);

    for (i = 0; i < argc; ++i) {
        printf("argv[%d]: %s\n", i, argv[i]);
    }

    return 0;
}
```

```
> a.exe < hello.txt
argc: 1
argv[0]: a.exe
```