6주차 1차시 프로세스 실행

【학습목표】

1. 리눅스 프로세스 생성을 설명할 수 있다.
2. 리눅스 프로세스 생성 함수를 활용할 수 있다.

학습내용1 : 리눅스 프로세스 생성

1. 프로세스 생성

; 프로그램 안에서 다른 프로그램을 실행해 새로운 프로세스를 생성하는 것

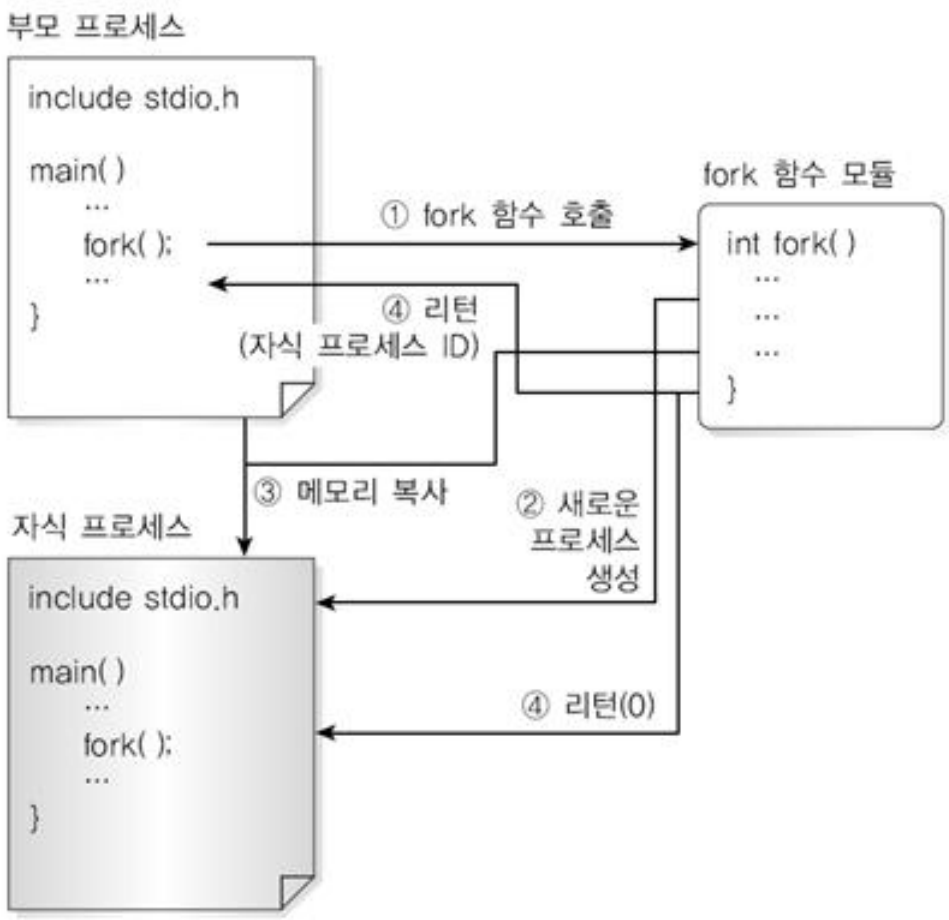
① 프로세스 생성 함수

기능	함수 원형
프로그램 실행	int system(const char *string);
프로세스 생성	pid_t fork(void); pid_t vfork(void);

② fork() 함수
새로운 프로세스를 생성 : 자식 프로세스
fork 함수를 호출한 프로세스 : 부모 프로세스
자식 프로세스는 부모 프로세스의 메모리를 복사
자식 프로세스 상속받는 속성

- RUID, EUID, RGID, EGID
- 환경변수
- 열린 파일기술자
- 시그널 처리
- setuid, setgid
- 현재 작업 디렉토리
- umask
- 사용가능자원 제한

* 부모 프로세스와 다른 점
- 자식 프로세스는 유일한 PID를 갖는다
- 자식 프로세스는 유일한 PPID를 갖는다.
- 부모 프로세스가 설정한 프로세스잠금, 파일 잠금, 기타 메모리 잠금은 상속 안함
- 자식 프로세스의 tms구조체 값은 0으로 설정
- 부모 프로세스와 자식 프로세스는 열린 파일을 공유하므로 읽거나 쓸 때 주의해야 한다.



③ 프로세스 종료 함수

기능	함수 원형
프로세스 종료	void exit(int status); void _exit(int status);
종료 시 수행할 작업 지정	int atexti(void (*func)(void));

정상 종료시 0 리턴, 실패시 0이 아닌값

④ exec 함수군

인자로 받은 다른 프로그램을 자신을 호출한 프로세스의 메모리에 덮어씀

프로세스가 수행중이던 기존 프로그램 중지되고 새로 덮어쓴 프로그램 실행

fork 함수와 연결해 fork 생성한 자식 프로세스가 새로운 프로그램을 실행하도록 함

기능	함수 원형

프로그램 실행	int execl(const char *path, const char *arg0, ..., const char *argn, (char *)0);
	int execv(const char *path, char *const argv[]);
	int execle(const char *path, const char *arg0, ..., const char *argn, (char *)0, char *const envp[]);
	int execve(const char *path, char *const argv[], char *const envp[]);
	int execlp(const char *file, const char *arg0, ..., const char *argn, (char *)0);
	int execvp(const char *file, char *const argv[]);

⑤ 프로세스 동기화

* 좀비 프로세스 방지

기능	함수 원형
임의의 자식 프로세스의 상태값 구하기	pid_t wait(int *stat_loc);
특정 프로세스의 상태값 구하기	pit_t waitpid(pid_t pid, int *stat_loc, int options);

학습내용2 : 리눅스 프로세스 생성 함수

1. 프로세스 생성

① 프로그램 실행 : system(3)
프로그램 안에서 새로운 프로그램 실행
쉘까지 동작시키므로 비효율적

```
#include <stdlib.h>
int system(const char *string);
```

string : 실행할 명령이나 실행 파일명

```
01  #include <stdlib.h>
02  #include <stdio.h>
03
04  int main(void) {
05     int a;
06     a = system("ps -ef | grep han > han.txt");
07     printf("Return Value : %d\n", a);
08
09     return 0;
10  }
```

```
# ex6_1.out
Return Value : 0
```

```
# cat han.txt
root 736 735 0 10:31:02 pts/3  0:00 grep han
root 735 734 0 10:31:02 pts/3  0:00 sh -c ps -ef | grep han> han.txt
```

② fork(2)

새로운 자식 프로세스 생성

성공 시 부로는 자식 프로세스의 ID 리턴, 자식 프로세스는 0 리턴

실패 시 -1 리턴

```
#include <sys/types.h>
#include <unistd.h>
pid_t fork(void);
```

인자를 받지 않음

```
...
06  int main(void) {
07      pid_t pid;
08
09      switch (pid = fork()) {
10          case -1 :  /* fork failed */
11              perror("fork");
12              exit(1);
13              break;
14          case 0 :   /* child process */
15          printf("Child Process - My PID:%d, My Parent's PID:%d\n",
16                  (int)getpid(), (int)getppid());
17              break;
18          default : /* parent process */
19          printf("Parent process - My PID:%d, My Parent's PID:%d, "
                "My Child's PID:%d\n", (int)getpid(), (int)getppid(),
                (int)pid);
21              break;
22      }
23
24      pri
26      ret
27  }
```

fork함수의 리턴값 0은 자식 프로세스가 실행

```
# ex6_2.out
Child Process - My PID:796, My Parent's PID:795
End of fork
Parent process - My PID:795, My Parent's PID:695, My Child's PID:796
End of fork
```

【학습정리】

1. 리눅스 프로세스 생성
- 프로그램 안에서 다른 프로그램을 실행해 새로운 프로세스를 생성하는 것

- 프로세스 생성 함수
① 프로그램 실행
int system(const char *string);
② 프로세스 생성
pid_t fork(void);
pid_t vfork(void);

- exec 함수군
① 인자로 받은 다른 프로그램을 자신을 호출한 프로세스의 메모리에 덮어씀 ② 프로세스가 수행중이던 기존 프로그램 중지되고 새로 덮어쓴 프로그램 실행
③ fork 함수와 연결해 fork 생성한 자식 프로세스가 새로운 프로그램을 실행하도록 함

2. 리눅스 프로세스 생성 함수
- 프로그램 실행 : system(3) : 프로그램 안에서 새로운 프로그램 실행

- fork(2)
① 새로운 자식 프로세스 생성
② 성공 시 부모는 자식 프로세스의 ID 리턴, 자식 프로세스는 0 리턴, 실패 시 -1 리턴