6주차 2차시 프로세스 종료

【학습목표】

1. 리눅스 프로세스 종료에 대해 설명할 수 있다.
2. 리눅스 프로세스 종료 함수를 활용할 수 있다.

학습내용1 : 리눅스 프로세스 종료

1. 프로세스의 종료

① 프로그램 종료 함수의 일반적 종료 절차
모든 파일 기술자를 닫는다.
부모 프로세스에 종료 상태를 알린다.
자식 프로세스들에 SIGHUP 시그널을 보낸다.
부모 프로세스에 SIGCHLD 시그널을 보낸다.
프로세스간 통신에 사용한 자원을 반납한다.

② 프로세스 종료 특징
프로세스가 마지막 명령 실행, 종료하여 운영체제에 프로세스의 삭제 요청
abort 명령어로 프로세스 종료

* 부모 프로세스의 자식 프로세스 종료
- 보통 부모 프로세스 종료하면 운영체제가 자식 프로세스도 종료(연속 종료)
- 자식 프로세스가 할당된 자원을 초과하여 자원을 사용할 때
- 자식 프로세스에 할당한 작업이 더는 없을 때

exit 명령어 : 유닉스에서 프로세스 종료
wait 명령어 : 부모 프로세스가 자식 프로세스의 종료 기다림.

* 프로세스 종료 이유
- 정상 종료 : 프로세스가 운영체제의 서비스 호출
- 시간 초과 :
- 실패 : 파일 검색 실패, 입출력이 명시된 횟수 초과하여 실패할 때
- 산술 오류, 보호 오류, 데이터 오류 등, 메모리 부족, 액세스 위반 등

학습내용2 : 리눅스 프로세스 종료 함수

1. 프로그램 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));

① 프로그램 종료: exit(2)

```
#include <stdlib.h>
void exit(int status);
```

status : 종료 상태값

② 프로그램 종료시 수행할 작업 예약: atexit(2)

```
#include <stdlib.h>
int atexit(void (*func)(void));
```

func : 종료시 수행할 작업을 지정한 함수명

③ 프로그램 종료: _exit(2)

```
#include <unistd.h>
void _exit(int status);
```

일반적으로 프로그램에서 직접 사용하지 않고 exit 함수 내부적으로 호출

```
01  #include <stdlib.h>
02  #include <stdio.h>
03
04  void cleanup1(void) {
05      printf("Cleanup 1 is called.\n");
06  }
07
08  void cleanup2(void) {
09      printf("Cleanup 2 is called.\n");
10  }
11
12  int main(void) {
13      atexit(cleanup1);
14      atexit(cleanup2);
15
16      exit(0);
17  }
```

종료시 수행할 함수 지정
지정한 순서의 역순으로 실행
(실행결과 확인)

```
# ex6_3.out
Cleanup 2 is called.
Cleanup 1 is called.
```

【학습정리】

1. 리눅스 프로세스 종료
- 프로그램 종료 함수의 일반적 종료 절차
- 모든 파일 기술자를 닫는다.
- 부모 프로세스에 종료 상태를 알린다.
- 자식 프로세스들에 SIGHUP 시그널을 보낸다.
- 부모 프로세스에 SIGCHLD 시그널을 보낸다.
- 프로세스간 통신에 사용한 자원을 반납한다.

2. 리눅스 프로세스 종료 함수
- 프로세스 종료

void exit(int status);
void _exit(int status);

- 종료 시 수행할 작업 지정

int atexti(void (*func)(void));