

## 1. multitasking program 작성

각 운영체제는 프로세스(부모)가 프로세스(자식)을 생성할 때, 두 프로세스를 실행시키는 방법이 있다.

첫 번째. 부모는 자식과 병행하게 실행을 계속한다. -> 부모는 일부 또는 모든 자식이 실행을 종료할 때까지 기다린다. -> 리눅스

두 번째. 자식 프로세스는 부모 프로세스의 복사본이다. -> 자식 프로세스가 자신(부모)에게 적재될 새로운 프로그램을 갖고 있다. -> 윈도우

### 리눅스

부모프로세스는 fork()를 호출하며, 이는 pid에 저장된다.

부모프로세스와 자식프로세스는 동일한 프로그램이다. 이 동일한 프로그램의 차이점은 자식 프로세스에게 보이는 pid값은 0이고 부모는 0보다 큰 정수값이다.

자식프로세스는 파일 데이터뿐 아니라 특권과 스케줄링 속성을 부모 프로세스로부터 상속받는다. 그 후 시스템호출을 사용해서 자신의 주소공간을 unix 명령 /bin/ls로 덮어쓴다.

부모는 wait() 호출로 자식 프로세스가 멈출 때까지 기다린다. 호출 후 exit() 시스템 호출을 사용하여 끝낸다.

### 윈도우

CreateProcess()는 자식 프로세스가 생성될 때 주소 공간에 명시된 프로그램을 적재한다. 이때 10개 이상의 매개변수가 필요하다.

이 프로그램에서는 많은 수의 매개변수를 디폴트로 선택하여 실행한다.

사진속 코드만 가지고는 실행되지 않아서, 헤더파일 두 개를 추가했다.

HW3-1\_linux.c

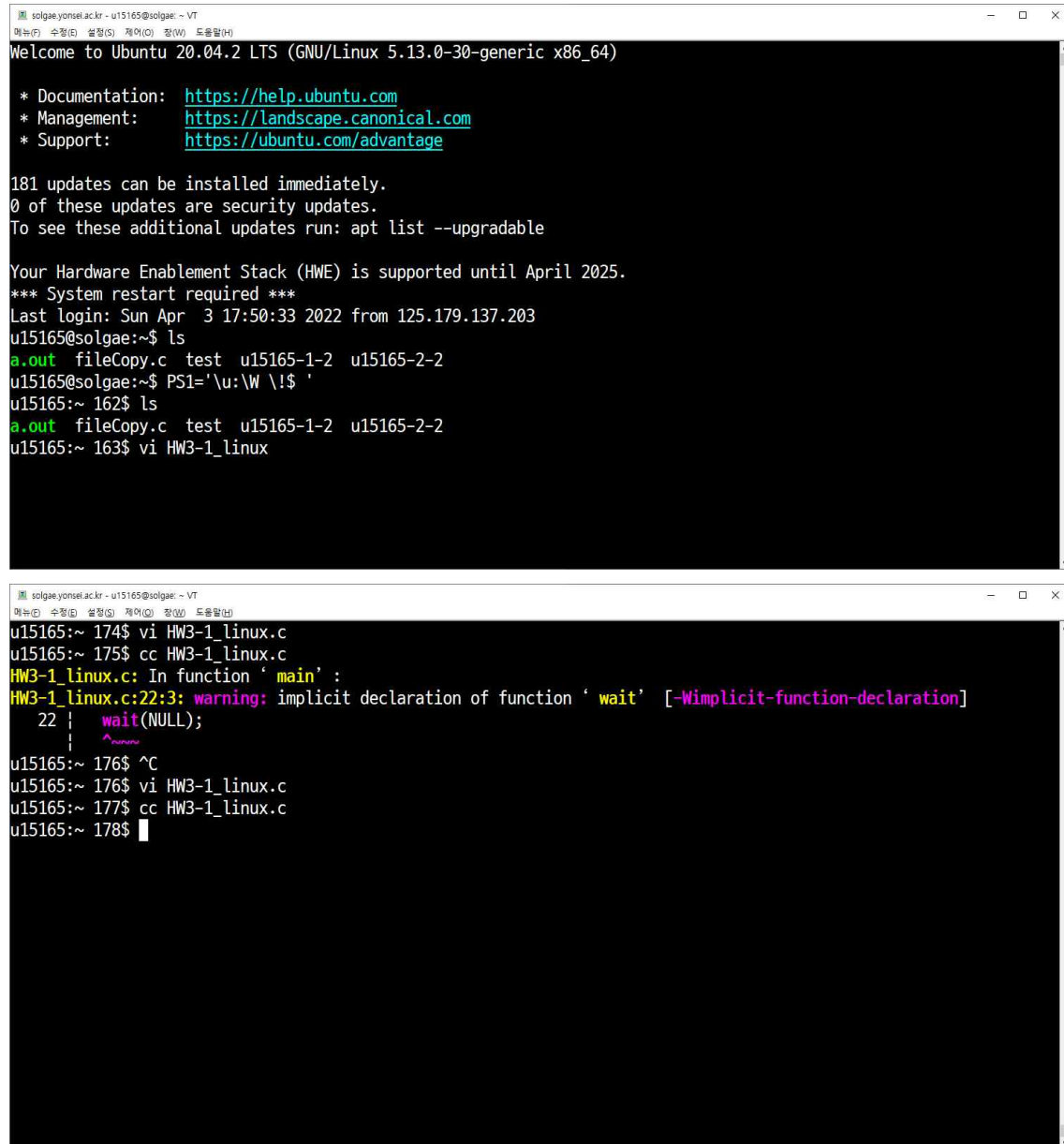
```
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <stdlib.h>
#include <sys/wait.h> // 헤더 추가
int main()
{
    pid_t pid;

    /* fork a child process */
    pid = fork();
    if (pid < 0) { /* error occurred */
        fprintf(stderr, "Fork Failed\n");
        exit(-1);
    }
    else if (pid == 0) { /* child process */
        printf("I am the child %d\n", pid);
        execlp("/bin/ls", "ls", NULL);
    }
    else { /* parent process */
        /* parent will wait for the child to complete */
        printf("I am the parent %d\n", pid);
        wait(NULL);

        printf("Child Complete\n");
        exit(0);
    }
}
```

HW3-1\_windows.c는 fig 3-11과 동일하다.

tera term에서 실행한 모습



```
solgae.yonse.ac.kr - u15165@solgae: ~ - VT
메뉴(M) 수정(E) 설정(S) 제어(C) 창(W) 도움말(H)

Welcome to Ubuntu 20.04.2 LTS (GNU/Linux 5.13.0-30-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

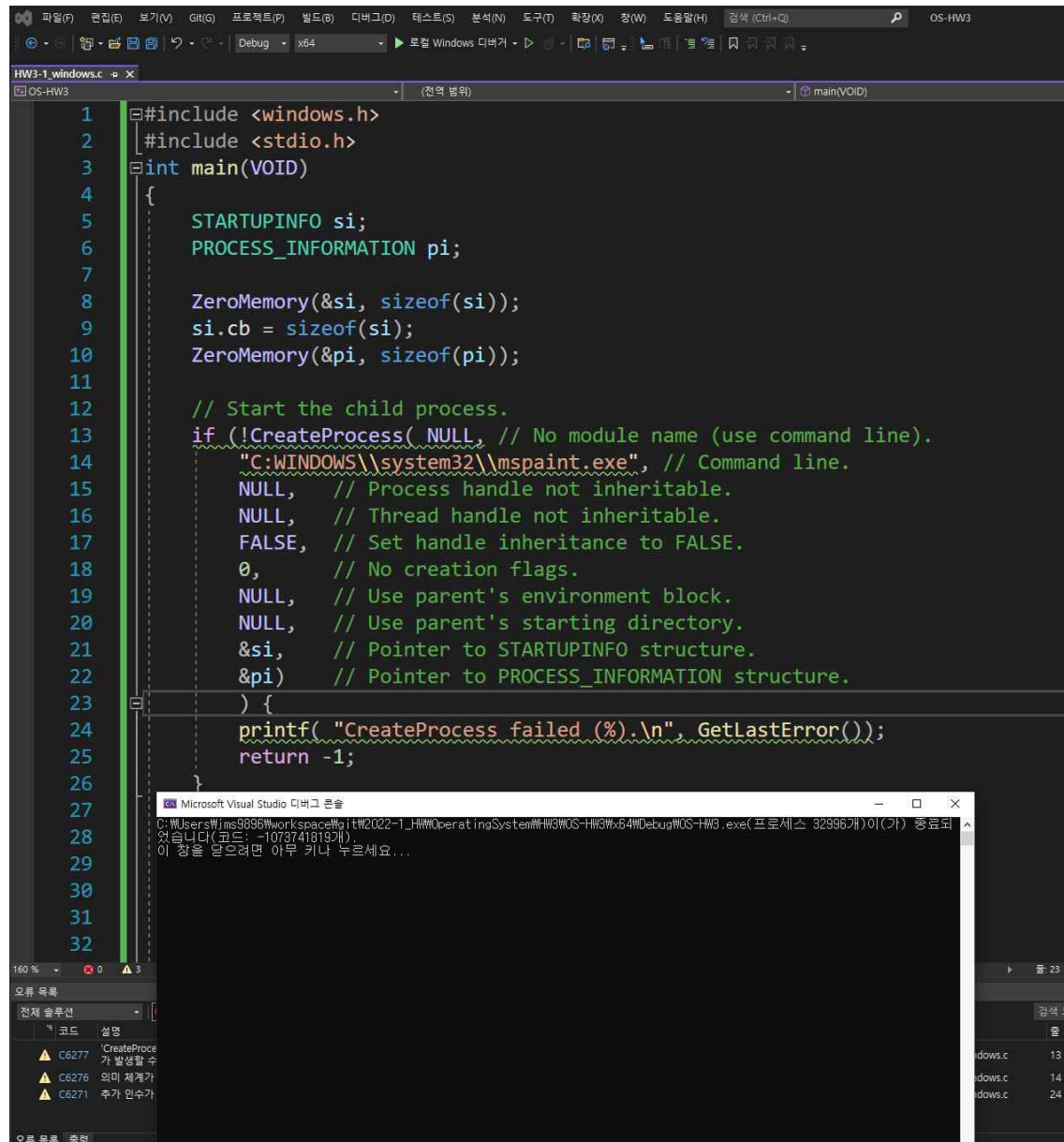
181 updates can be installed immediately.
0 of these updates are security updates.
To see these additional updates run: apt list --upgradable

Your Hardware Enablement Stack (HWE) is supported until April 2025.
*** System restart required ***
Last login: Sun Apr  3 17:50:33 2022 from 125.179.137.203
u15165@solgae:~$ ls
a.out  fileCopy.c  test  u15165-1-2  u15165-2-2
u15165@solgae:~$ PS1='\u:\W \!$ '
u15165:~ 162$ ls
a.out  fileCopy.c  test  u15165-1-2  u15165-2-2
u15165:~ 163$ vi HW3-1_linux

u15165:~ 174$ vi HW3-1_linux.c
u15165:~ 175$ cc HW3-1_linux.c
HW3-1_linux.c: In function 'main' :
HW3-1_linux.c:22:3: warning: implicit declaration of function 'wait' [-Wimplicit-function-declaration]
   22 |     wait(NULL);
      |     ^~~~~
u15165:~ 176$ ^C
u15165:~ 176$ vi HW3-1_linux.c
u15165:~ 177$ cc HW3-1_linux.c
u15165:~ 178$
```

결과물이 보이지 않는다.

visual studio에서 실행한 모습



```
1 #include <windows.h>
2 #include <stdio.h>
3 int main(VOID)
4 {
5     STARTUPINFO si;
6     PROCESS_INFORMATION pi;
7
8     ZeroMemory(&si, sizeof(si));
9     si.cb = sizeof(si);
10    ZeroMemory(&pi, sizeof(pi));
11
12    // Start the child process.
13    if (!CreateProcess( NULL, // No module name (use command line).
14        "C:\WINDOWS\system32\cmd.exe", // Command line.
15        NULL, // Process handle not inheritable.
16        NULL, // Thread handle not inheritable.
17        FALSE, // Set handle inheritance to FALSE.
18        0, // No creation flags.
19        NULL, // Use parent's environment block.
20        NULL, // Use parent's starting directory.
21        &si, // Pointer to STARTUPINFO structure.
22        &pi) // Pointer to PROCESS_INFORMATION structure.
23    ) {
24        printf( "CreateProcess failed (%d).\n", GetLastError());
25        return -1;
26    }
27
28
29
30
31
32
```

Microsoft Visual Studio 디버그 콘솔

C:\Users\W\ms9896\workspace\git\2022-1\_HW\OperatingSystem\HW3\OS-HW3\Debug\OS-HW3.exe( 프로세스 32996개)이(가) 종료되었습니다(코드: -1073741819개).  
이 창을 알으려면 아무 키나 누르세요...

오류 목록

전체 솔루션	코드	설명
▲ C6277		'CreateProce가 발생할 수
▲ C6276		의미 체계가
▲ C6271		추가 인수가

오류 목록 출력

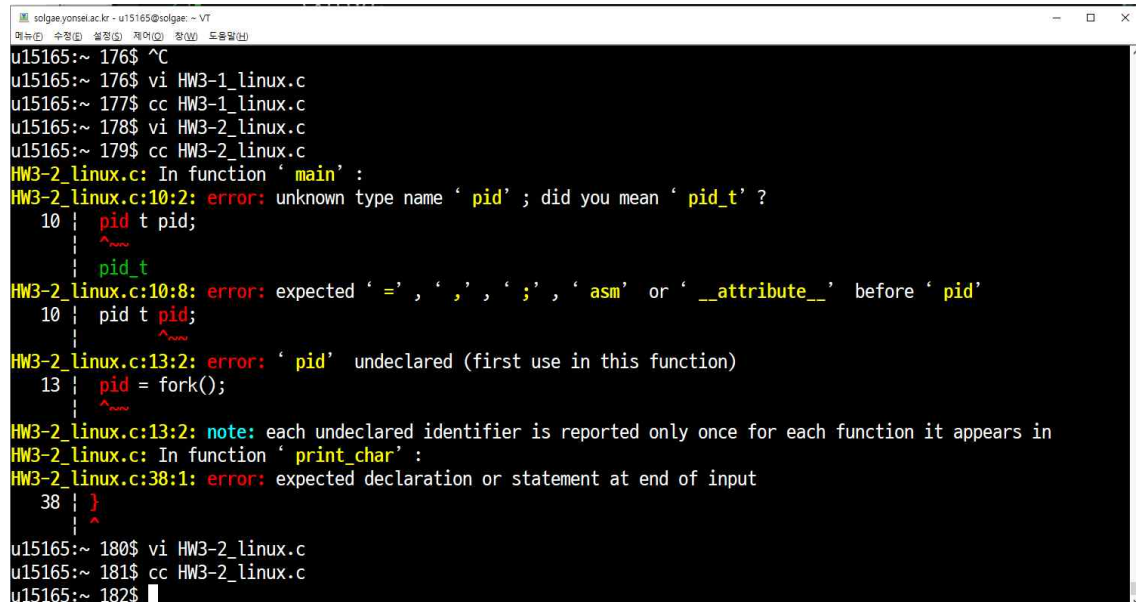
검색	줄
ows.c	13
ows.c	14
ows.c	24

결과물이 보이지 않는다.

## 2. 문자 출력을 수행하는 두 concurrent 프로그램 만들기

실행 결과 출력

HW3-2\_linux.c



```
u15165:~ 176$ ^C
u15165:~ 176$ vi HW3-1_linux.c
u15165:~ 177$ cc HW3-1_linux.c
u15165:~ 178$ vi HW3-2_linux.c
u15165:~ 179$ cc HW3-2_linux.c
HW3-2_linux.c: In function 'main':
HW3-2_linux.c:10:2: error: unknown type name 'pid'; did you mean 'pid_t'?
   10 | pid t pid;
      |      ^
      |      pid_t
HW3-2_linux.c:10:8: error: expected '=', ',', ';', 'asm' or '__attribute__' before 'pid'
   10 | pid t pid;
      |        ^
HW3-2_linux.c:13:2: error: 'pid' undeclared (first use in this function)
   13 | pid = fork();
      |      ^
HW3-2_linux.c:13:2: note: each undeclared identifier is reported only once for each function it appears in
HW3-2_linux.c: In function 'print_char':
HW3-2_linux.c:38:1: error: expected declaration or statement at end of input
   38 | }
      | ^
u15165:~ 180$ vi HW3-2_linux.c
u15165:~ 181$ cc HW3-2_linux.c
u15165:~ 182$
```

결과물이 보이지 않는다.

소스에 대한 동작 설명

pid가 0이면 자식프로세스이며, '-'가 100번 출력된다. 그 후 stdout버퍼를 flushing한다.

for문을 통해 천 번 delay 후 exit(0)를 반환한다.

pid가 0보다 큰 정수이면 0을 반환하고 flushing한다.

## 3. execl, execlp, execl, execv, execvp, execvpe의 인수들이 어떠한 차이가 있는지 비교 설명하시오.

exec 함수 계열은 현재 프로세스 이미지를 새로운 프로세스 이미지로 바꾸는 작업들이다.

int execl(const char \*path, const char \*arg0, ..., const char \*argn, (char \*)0);  
path에 지정한 경로 명의 파일을 실행하며 arg0~argn을 인자로 전달한다. (관례적으로 arg0에는 실행 파일명을 지정한다.)

int execv(const char \*path, char \*const argv[] );

path에 지정한 경로명에 있는 파일을 실행하며 argv를 인자로 전달한다.

int execl(const char \*path, const char \*arg0, ..., const char \*argn, (char \*)0, char \*const envp[]);

path에 지정한 경로명의 파일을 실행하며 arg0 argn과 envp 를 인자로 전달한다.

int execve(const char \*path, char \*const argv[], char \*const envp[]);

path에 지정한 경로 명의 파일을 실행하며 argv, envp 를 인자로 전달한다.

```
int execlp(const char *file, const char *arg0, ...,const char *argn, (char *)0);
```

file에 지정한 파일을 실행하며 argorargn 만 인자로 전달한다. 파일은 이 함수를 호출한 프로세스의 검색 경로 (환경 변수 PATH에 정의된 경로)에서 찾는다.

```
int execvp(const char *file, char *const argv[]);
```

file에 지정한 파일을 실행하며 argv를 인자로 전달한다.

4. I/O bound process와 CPU bound process란 무엇인가? 그리고 long term scheduler의 역할은 무엇이며 어떻게 메모리에 적재할 프로세스를 선택하는가?

I/O bound process

프로그램 실행이 I/O에 의존하는 프로세스다.

입출력 시스템을 통해 리소스를 요청하므로, 이 작업이 빨라진다면 프로그램이 빨라지는 효과를 가져올 수 있다.

CPU를 덜 사용하거나 사용하지 않을 수 있다.

CPU bound process

작업이나 프로그램의 실행이 CPU에 크게 의존하는 프로세스이다.

CPU bound process에서는 CPU가 프로그램 실행에 대부분 관여한다. 프로그램 실행 속도를 높이려면 CPU 성능을 높여야한다.

CPU를 사용해서 작업하는 시간이 오래 걸릴 수 있으므로, CPU bound process를 줄이는 것이 좋다.

5. 프로세스가 Running 상태에서 Waiting 상태로 전이되는 예를 여러분의 프로그램 경험을 토대로 몇 가지만 제시해보시오.

대표적으로 ctrl + alt + delete를 누르게 되면 작업중에 중단이 가능하다.

CLI의 경우 ctrl + c를 눌러 작업을 중지시킬 수도 있다.

visual studio의 경우 pause break를 눌러 작업을 중지시킨적도 있다.

6. UNIX/Linux에서 고아 프로세스의 새로운 부모로 어떤 프로세스가 지정되는가? 그리고 init 프로세스가 고아 프로세스에 대해서 하는 역할은 무엇인가?

init process가 지정된다.

init process는 기존 process의 잔해(찌꺼기) 처리를 담당한다.

7. interprocess communicatio(IPC)를 구현하는 두가지 기본 모델을 적고, 두 모델이 통신하는 방법을 간단히 설명하시오.

shared memory

message passing

shared memory(공유메모리)방식에서는 협력하는 프로세스가 공유하는 메모리 영역을 설정한다. 그러면 프로세스는 공유메모리의 데이터를 읽고 쓰는 방식으로 정보를 교환한다.

프로세스가 다른 프로세스에게 정보를 줄 때 memory protection이 일어나는데, message passing 방식은 kernel의 도움을 받아서 정보를 공유할 수 있다.

한 프로세서는 커널에 데이터를 전송하고, 커널은 다른 프로세스에게 데이터를 전송할 수 있다.