운영체제 과제 - 3장

※ 다음 과제에서 UNIX/Linux 시스템을 사용할 때에는 로그인 직후에 다음 명령어를 실행하여 프롬프트를 변경한 후 사용하며 동작 결과를 보여줄 때에 프롬프트도 함께 보여주시오.

```
PS1='\u:\W \!$ '
```

1. **(multitasking program 작성)** 교과서 3장의 **Figure 3-9** (UNIX/Linux 프로그램)과 **Figure 3-11** (Windows 프로그램)에 있는 프로그램을 Linux(magics) 와 Windows 운영체제 환경에서 각각 작성하여 컴파일하고 실행시켜서 동작을 관찰하시오. 그리고 이 프로그램들에서 사용한 운영체제 API의 사용법을 조사하고 동작에 대해서 설명하시오.

Fig 3-9

```c
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>

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);
        }
}
```

Fig 3-11

```c
#include <windows.h>
#include <stdio.h>

int main( VOID )
{
    STARTUPINFO si;
    PROCESS_INFORMATION pi;

    ZeroMemory( &si, sizeof(si) );
    si.cb = sizeof(si);
    ZeroMemory( &pi, sizeof(pi) );

    // Start the child process.
    if( !CreateProcess( NULL,   // No module name (use command line).
        "C:\\WINDOWS\\system32\\mspaint.exe", // Command line.
        NULL,             // Process handle not inheritable.
        NULL,             // Thread handle not inheritable.
        FALSE,            // Set handle inheritance to FALSE.
        0,                // No creation flags.
        NULL,             // Use parent's environment block.
        NULL,             // Use parent's starting directory.
        &si,              // Pointer to STARTUPINFO structure.
        &pi )             // Pointer to PROCESS_INFORMATION structure.
    )
    {
        printf( "CreateProcess failed (%d).\n", GetLastError() );
        return -1;
    }

    // Wait until child process exits.
    WaitForSingleObject( pi.hProcess, INFINITE );

    // Close process and thread handles.
    CloseHandle( pi.hProcess );
    CloseHandle( pi.hThread );
}
```

2. unix/linux에서 제공하는 exec 계열의 시스템 호출 함수들을 모두 적고 인수들이 어떠한 차이가 있는지 비교 설명하시오. (이에 대한 내용은 홈페이지의 [process] 참고자료와 인터넷 검색 자료를 참고하시오.)

3. (문자출력을 수행하는 두 concurrent 프로그램 만들기) Unix/Linux에서 다음 프로그램을 작성하여 실행시키시오. 그리고 프로그램 소스에 대한 동작 설명, 실행 결과 출력 및 토의를 함께 적으시오.

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>

void print_char(int ch);

int main()
{
        pid_t pid;
        int i, j;

        /* fork a child process */
        pid = fork();

        if (pid < 0) { /* error occurred */
                fprintf(stderr, "Fork Failed\n");
                exit(-1);
        } else if (pid == 0) { /* child process */
                print_char('-');
                exit(0);
        } else { /* parent process */
                print_char('O');
                exit(0);
        }
}
```

```c
void print_char(int ch)
{
        int i, j;

        for (i=0; i<100; i++)
        {
                putchar(ch);
                fflush(stdout);    /* flush stdout buffer */
                for (j=0; j<1000; j++)  /* delay */
                        ;
        }
}
```

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

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

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

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