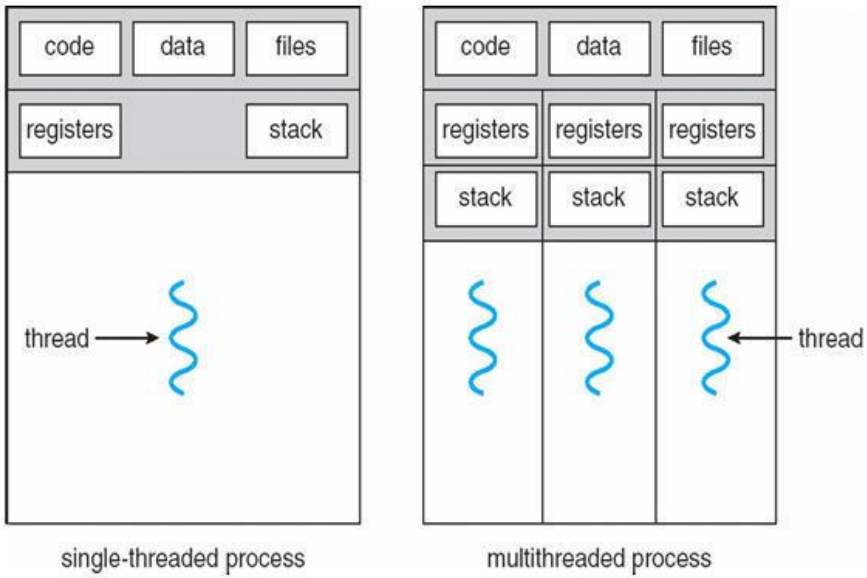
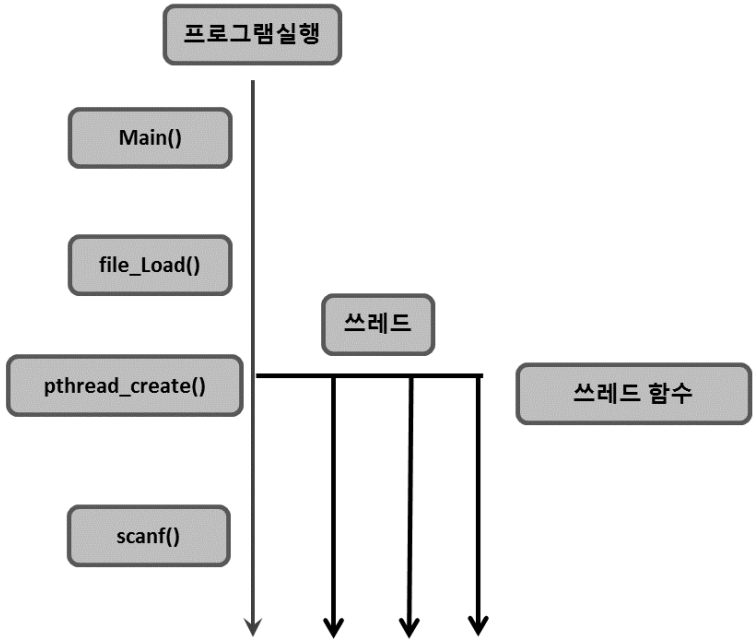
Thread

Thread





자료 출처 : Operating System Concepts (Abraham Silberschatz)

Thread

- int pthread_create(pthread_t * thread, pthread_att_t *attr, void * (*start_routine)(void *), void *arg)
  - thread : 생성된 스레드를 식별하기 위한 쓰레드 식별자 번호
  - attr : 생성되는 스레드의 특성을 정의하기 위해 사용(기본 특성 : NULL)
  - start_routine : 만들고자 하는 스레드 함수의 포인터(실행될 스레드 함수)
  - arg : 스레드 함수(start_routine)가 실행될 때 넘어갈 매개변수

~/personal_project/thirdProject

```
 1 #include <stdio.h>
 2 #include <pthread.h>
 3
 4 pthread_t thread;
 5
 6 void *thread_function(void *);
 7
 8 int main(void){
 9     int i = 10;
10     printf("입 력  : ");
11     scanf("%d", &i);
12     pthread_create(&thread, NULL, &thread_function, (void *)i);
13     return 0;
14 }
15
16 void *thread_function(void *arg){
17     int i = (int)arg;
18     printf("i = %d\n", i);
19     return ;
20 }
21
~
~
~
                                                              1,5
```

~/personal_project/thirdProject

```
DSLab@DESKTOP-10JODED ~/personal_project/thirdProject
$ ls
a.exe  a.exe.stackdump  mirror.c  test.c  thirdProject.zip

DSLab@DESKTOP-10JODED ~/personal_project/thirdProject
$ ./a.exe
입 력  : 10
i = 10

DSLab@DESKTOP-10JODED ~/personal_project/thirdProject
$
```

Thread

- int pthread_join(pthread_t thread, void **thread_return)
  - 스레드의 종료를 기다려 스레드의 종료 값을 받아오고 스레드의 자원을 정리하는 함수
  - thread : join하고자 하는 스레드의 식별자
  - thread_return : 스레드의 반환 값

~/personal_project/thirdProject/thread_test

```
 1 #include <stdio.h>
 2 #include <pthread.h>
 3
 4 pthread_t thread;
 5
 6 void *thread_function(void *);
 7
 8 int main(void){
 9     int i = 10;
10     pthread_create(&thread, NULL, &thread_function, (void *)i);
11     for(int j = 0; j<i; j++){
12         printf("main, j = %d \n", j);
13     }
14     pthread_join(thread, NULL);
15
16     return 0;
17 }
18
19 void *thread_function(void *arg){
20     int i = (int)arg;
21     printf("i = %d\n", i);
22     for(int j = 0; j<i; j++){
23         printf("thread, j = %d \n", j);
24     }
25     return ;
26 }
"test2.c" 27L, 442C                                   23,3-9        꼭 대 기
```

~/personal_project/thirdProject/thread_test

```
DSLab@DESKTOP-10J0DED ~/personal_project/thirdProject/thread_test
$ ./test2.exe
main, j = 0
i = 10
main, j = 1
thread, j = 0
main, j = 2
thread, j = 1
main, j = 3
thread, j = 2
main, j = 4
thread, j = 3
main, j = 5
thread, j = 4
main, j = 6
thread, j = 5
main, j = 7
thread, j = 6
main, j = 8
thread, j = 7
main, j = 9
thread, j = 8
thread, j = 9

DSLab@DESKTOP-10J0DED ~/personal_project/thirdProject/thread_test
$
```

~/personal_project/thirdProject/thread_test

```
DSLab@DESKTOP-10J0DED ~/personal_project/thirdProject/thread_test
$ ./test2.exe
i = 10
main, j = 0
thread, j = 0
main, j = 1
thread, j = 1
main, j = 2
thread, j = 2
main, j = 3
thread, j = 3
main, j = 4
thread, j = 4
main, j = 5
thread, j = 5
main, j = 6
thread, j = 6
main, j = 7
thread, j = 7
main, j = 8
thread, j = 8
main, j = 9
thread, j = 9

DSLab@DESKTOP-10J0DED ~/personal_project/thirdProject/thread_test
$
```

gotoxy

- void gotoxy(int x, int y)
  - 터미널에서 커서의 위치를 제어 함수
  - 윈도우에서는 <windows.h> 에 함수가 존재
  - 리눅스에서는 gotoxy 함수가 없어서 직접 구현해야 함

~/personal_project/thirdProject/thread_test

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


void gotoxy(int x, int y);

int flag = 0;

int main(void){
    int i = 10;
    system("clear");
    while(1){
        gotoxy(0,0);
        printf("Input : ");
        scanf("%d", &flag);
        gotoxy(9, 2);
        printf("      ");
        switch(flag){
        case 1:
            gotoxy(0, 5*flag);
            printf("1번  입 력 ");
            break;
        case 2:
            gotoxy(0, 5*flag);
            printf("2번  입 력 ");
            break;
        case 3:
            system("clear");
        }
    }

    return 0;
}

void gotoxy(int x, int y){
    printf("\033[%d;%df", y, x);
    fflush(stdout);
}
```

1,9 모 두

~/personal_project/thirdProject/thread_test

```
Input : 1


1번  입 력
```

~/personal_project/thirdProject/thread_test

```
Input : 2


1번  입 력


2번  입 력
```