5주차_
멀티쓰레드

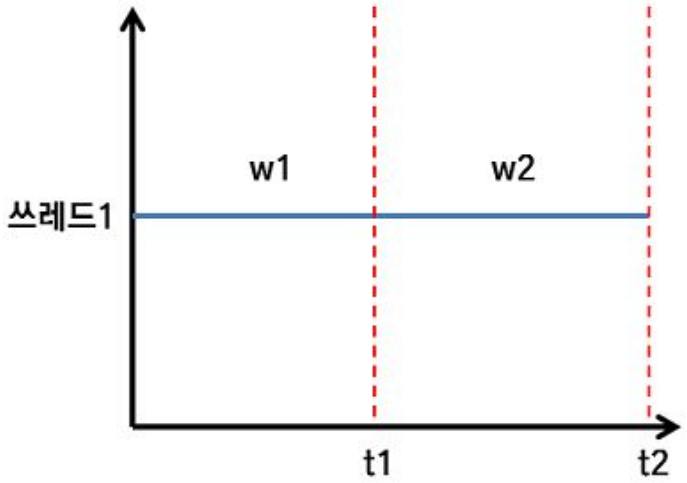
데이터 네트워크연구실
이현호
leeo75@cs-cnu.org

Goals

- 멀티쓰레드 이해하기
- 네트워크에서 멀티쓰레드

Thread

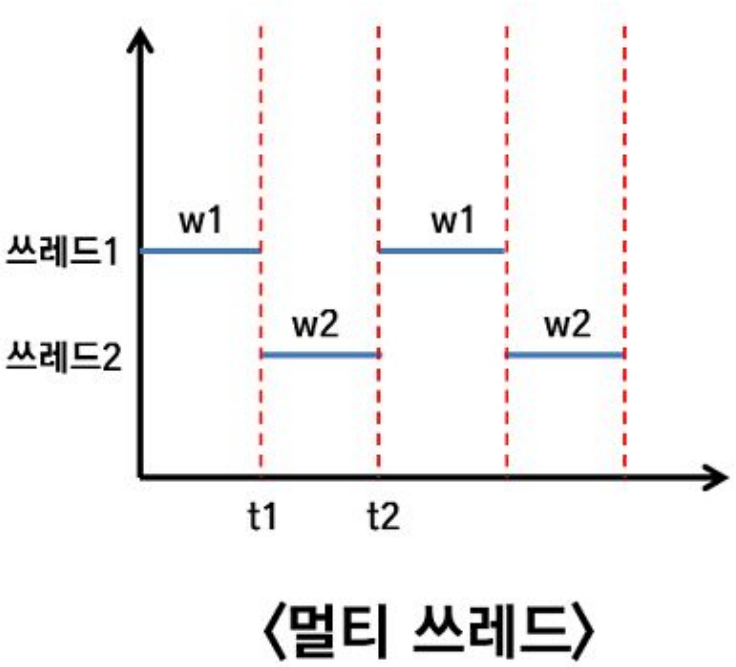
- 운영체제의 최소 실행단위는 프로세스
- 프로세스 내에서 실행단위



〈싱글 쓰레드〉

Multi Thread

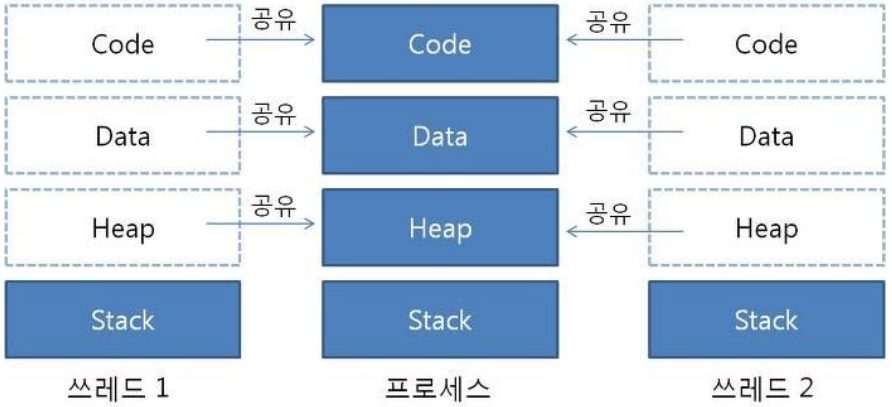
- 둘 이상의 스레드를 동시에 실행
- 스위칭 해 멀티 태스킹
- 프로세스의 메모리 공유



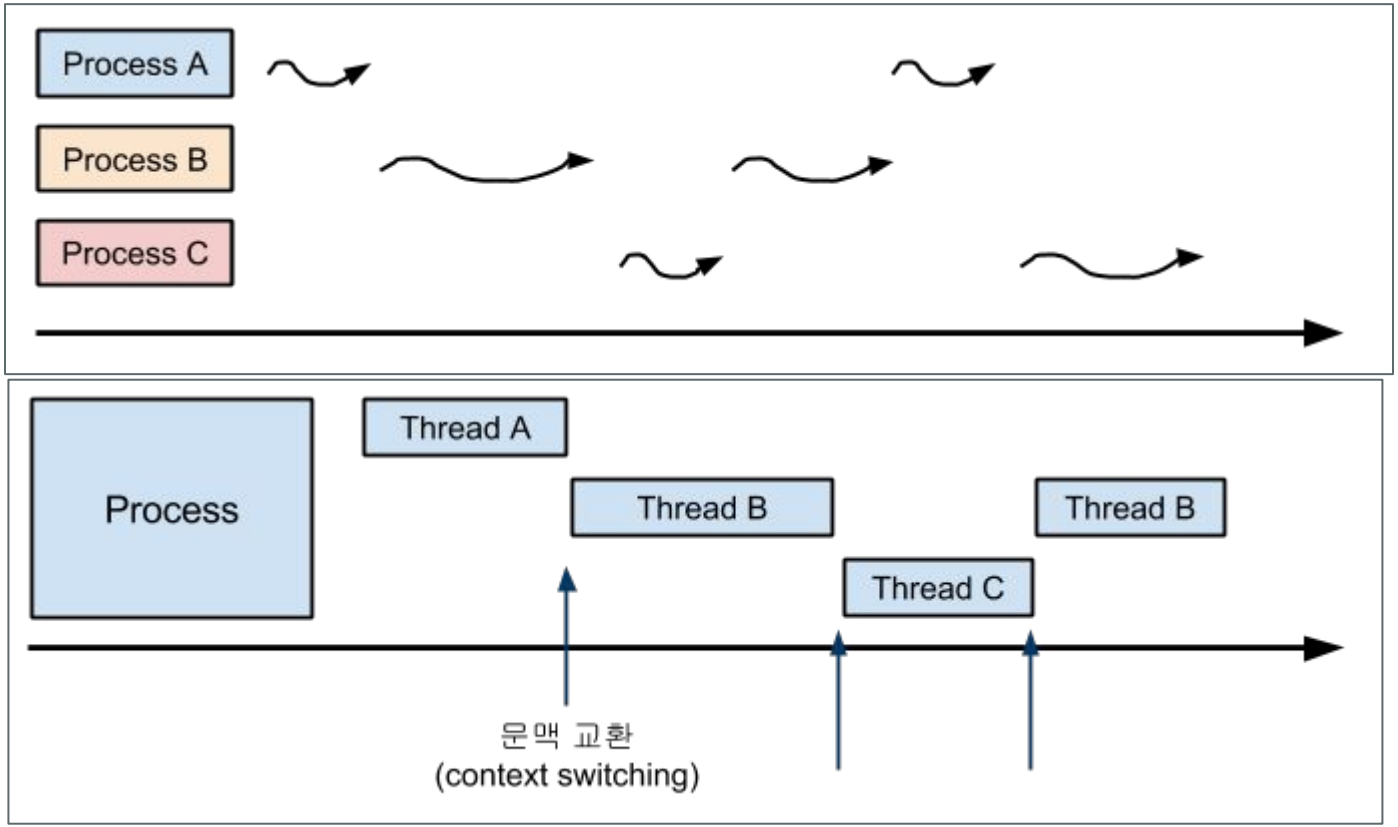
〈멀티 쓰레드〉

Thread and Process

- Code, Data, Heap, Stack 영역으로 이루어짐
- 프로세스 안에서 동작하며 Code, Data, Heap 영역을 공유하고 별도의 Stack영역을 가짐
- 쓰레드 간 자원 공유가 가능하여 편리하지만 자원 동기화의 문제



Thread and Process



Example

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

#define MAX_THREAD 2
void *t_func(void *data)
{
        int *count = (int *)data;
        int tmp;
        pthread_t thread_id = pthread_self();

        while(1)
        {
                printf("%lu %d\n", thread_id, *count);
                *count = *count+1;
                sleep(1);
        }
}
```

Example

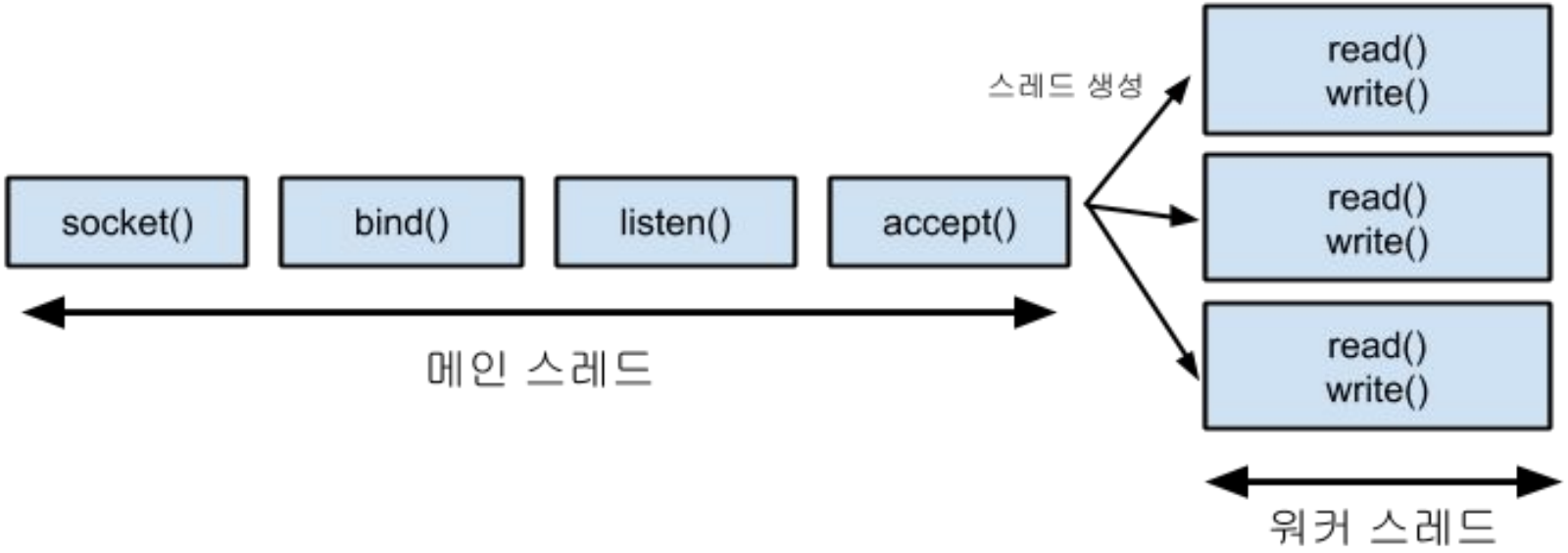
```
21 int main(int argc, char **argv)
22 {
23         pthread_t thread_id[MAX_THREAD];
24         int i = 0;
25         int count = 0;
26
27         for(i = 0; i < MAX_THREAD; i++)
28         {
29                 pthread_create(&thread_id[i], NULL, t_func, (void *)&count);
30                 usleep(5000);
31         }
32
33         while(1)
34         {
35                 printf("Main Thread : %d\n", count);
36                 sleep(2);
37         }
38         for(i = 0; i < MAX_THREAD; i++)
39         {
```

```
40                 pthread_join(thread_id[i], NULL);
41         }
42         return 0;
43 }
```

Example

- 컴파일이 안될 경우 pthread 포함 시켜줘야함
- gcc -pthread count_thread.c -o count_thread

Multi thread in network



Server

Code

```
 1 #include <pthread.h>
 2 #include <sys/types.h>
 3 #include <sys/socket.h>
 4
 5 #include <netinet/in.h>
 6 #include <arpa/inet.h>
 7 #include <stdio.h>
 8
 9 #include <stdlib.h>
10 #include <unistd.h>
11 #include <string.h>
12
13 #define MAXLINE 1024
14 #define PORTNUM 6292
15
16 void * thread_func(void *data)
17 {
18         int sockfd = *((int *)data);
19         int readn;
```

Code

```
20          socklen_t addrlen;
21          char buf[MAXLINE];
22          struct sockaddr_in client_addr;
23          memset(buf, 0x00, MAXLINE);
24          addrlen = sizeof(client_addr);
25          getpeername(sockfd, (struct sockaddr *)&client_addr, &addrlen);
26          while((readn = read(sockfd, buf, MAXLINE)) > 0)
27          {
28                  printf("Read Data %s(%d) : %s",
29                                  inet_ntoa(client_addr.sin_addr),
30                                  ntohs(client_addr.sin_port),
31                                   buf);
32                  write(sockfd, buf, strlen(buf));
33                  memset(buf, 0x00, MAXLINE);
34          }
35          close(sockfd);
36          printf("worker thread end\n");
37          return 0;
38  }
```

Code

```
39
40 int main(int argc, char **argv)
41 {
42         int listen_fd, client_fd;
43         socklen_t addrlen;
44         int readn;
45         char buf[MAXLINE];
46         pthread_t thread_id;
47
48         struct sockaddr_in server_addr, client_addr;
49
50         if( (listen_fd = socket(AF_INET, SOCK_STREAM, 0)) < 0)
51         {
52                 return 1;
53         }
54         memset((void *)&server_addr, 0x00, sizeof(server_addr));
55         server_addr.sin_family = AF_INET;
56         server_addr.sin_addr.s_addr = htonl(INADDR_ANY);
57         server_addr.sin_port = htons(PORTNUM);
```

Code

```
58
59          if(bind(listen_fd, (struct sockaddr *)&server_addr, sizeof(server_addr)) ==-1    )
60          {
61                  perror("bind error");
62                  return 1;
63          }
64          if(listen(listen_fd, 5) == -1)
65          {
66                  perror("listen error");
67                  return 1;
68          }
69
70          while(1)
71          {
72                  addrlen = sizeof(client_addr);
73                  client_fd = accept(listen_fd,
74                          (struct sockaddr *)&client_addr, &addrlen);
75                  if(client_fd == -1)
76                  {
```

Code

```
77                      printf("accept error\n");
78              }
79              else
80              {
81                      pthread_create(&thread_id, NULL, thread_func, (void *)&client_fd);
82                      pthread_detach(thread_id);
83              }
84      }
85      return 0;
86 }
87
88
```

Result

```
hyunholee@DNLAB:~/temp/Multi_thread$ ./echo_server_thread
Read Data 127.0.0.1(33134) : hi
Read Data 127.0.0.1(33134) : ho
Read Data 127.0.0.1(33134) : hi
Read Data 127.0.0.1(33134) : bye
Read Data 127.0.0.1(33142) : hihi
Read Data 127.0.0.1(33142) : bye
```

Result

```
hyunholee@DNLAB:~$ ps -aux | grep echo_server
hyunhol+ 16669  0.0  0.0  80256   716 pts/0    Sl+  08:08   0:00 ./echo_server_thread
hyunhol+ 16722  0.0  0.0  12944   980 pts/2    S+   08:09   0:00 grep --color=auto echo_server
hyunholee@DNLAB:~$ ps -p 16669 -T
  PID  SPID TTY          TIME CMD
16669 16669 pts/0    00:00:00 echo_server_thr
16669 16671 pts/0    00:00:00 echo_server_thr
hyunholee@DNLAB:~$ ps -p 16669 -T
  PID  SPID TTY          TIME CMD
16669 16669 pts/0    00:00:00 echo_server_thr
16669 16671 pts/0    00:00:00 echo_server_thr
16669 16779 pts/0    00:00:00 echo_server_thr
hyunholee@DNLAB:~$ ps -aux | grep echo_server
hyunhol+ 16669  0.0  0.0  88452   716 pts/0    Sl+  08:08   0:00 ./echo_server_thread
hyunhol+ 16795  0.0  0.0  12944   936 pts/2    S+   08:22   0:00 grep --color=auto echo_server
```

Upgrade

```
hyunholee@DNLAB:~/temp/Multi_thread$ ./echo_server_thread 6292
Read Data 127.0.0.1(33862) : hi
hello
Answer : Read Data 127.0.0.1(33864) : Man!
what's up?
Answer : Read Data 127.0.0.1(33864) : Find thanks
me too
Answer : Read Data 127.0.0.1(33864) : Are you linstening?
Yes
Answer : Read Data 127.0.0.1(33862) : hahaha
Why?
```

Upgrade

```
hyunholee@DNLAB:~
hi
send : hi
read : hello
hahaha
send : hahaha
read : Why?
```

```
hyunholee@DNLAB:~/temp/Mult
Man!
send : Man!
read : what's up?
Find thanks
send : Find thanks
read : me too
Are you linstening?
send : Are you linstening?
read : Yes
```