스레드 동기화 실습 및
리눅스 시스템 모니터링 시스템

임세진

https://youtu.be/hyfWdWMJ97Q

한성대학교 HANSUNG UNIVERSITY

CryptoCraft LAB
https://crypto.modoo.at

Contents

CryptoCraft LAB

01. 스레드 동기화 관련 실습

01. 스레드 동기화 관련 실습

- 스레드 동기화 없이 공유 데이터 사용했을 때 - 문제발생

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

int sum = 0; //share val

void* worker(void* arg){ //thread code
        for(int i=0; i<10000; i++){
                sum += 10;
        }
}

int main() {
        char *name[] = {"sejin", "runa"};
        pthread_t tid[2];
        pthread_attr_t attr[2];

        pthread_attr_init(&attr[0]);
        pthread_attr_init(&attr[1]);

        pthread_create(&tid[0], &attr[0], worker, name[0]); //create thread
        pthread_create(&tid[1], &attr[1], worker, name[1]);

        pthread_join(tid[0], NULL);
        pthread_join(tid[1], NULL);

        printf("total sum = %d \n", sum);

        return 0;
}
```

공유 데이터

문제 발생 원인 구간

스레드가 할 일을 마칠 때까지 기다림

01. 스레드 동기화 관련 실습

1. 상호 배제를 포함하는 프로그램 – 뮤텍스 (Mutex)

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

int sum = 0; //share val
pthread_mutex_t lock; //mutex lock

void* worker() {
        for(int i=0; i<10000; i++){
                pthread_mutex_lock(&lock); //entry code. lock
                sum += 10;
                pthread_mutex_unlock(&lock); //exit coide. unlock
        }
}
```

공유 데이터

상호 배제가 일어나는 구간

```c
int main(){
        char *name[] = {"sejin", "runa"};
        pthread_t tid[2];
        pthread_attr_t attr[2];

        pthread_attr_init(&attr[0]);
        pthread_attr_init(&attr[1]);

        pthread_mutex_init(&lock, NULL); //initialize mutex lock

        pthread_create(&tid[0], &attr[0], worker, name[0]); //create thread
        pthread_create(&tid[1], &attr[1], worker, name[1]);

        pthread_join(tid[0], NULL);
        pthread_join(tid[1], NULL);

        printf("total sum = %d \n", sum);

        pthread_mutex_destroy(&lock);

        return 0;
}
```

01. 스레드 동기화 관련 실습

2. 상호 배제를 포함하는 프로그램 – 스핀락 (Spinlock)

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

int sum = 0; //share val
pthread_spinlock_t lock; //spin lock

void* worker() {
        for(int i=0; i<10000; i++){
                pthread_spin_lock(&lock); //entry code. lock
                sum += 10;
                pthread_spin_unlock(&lock); //exit coide. unlock
        }
}
```

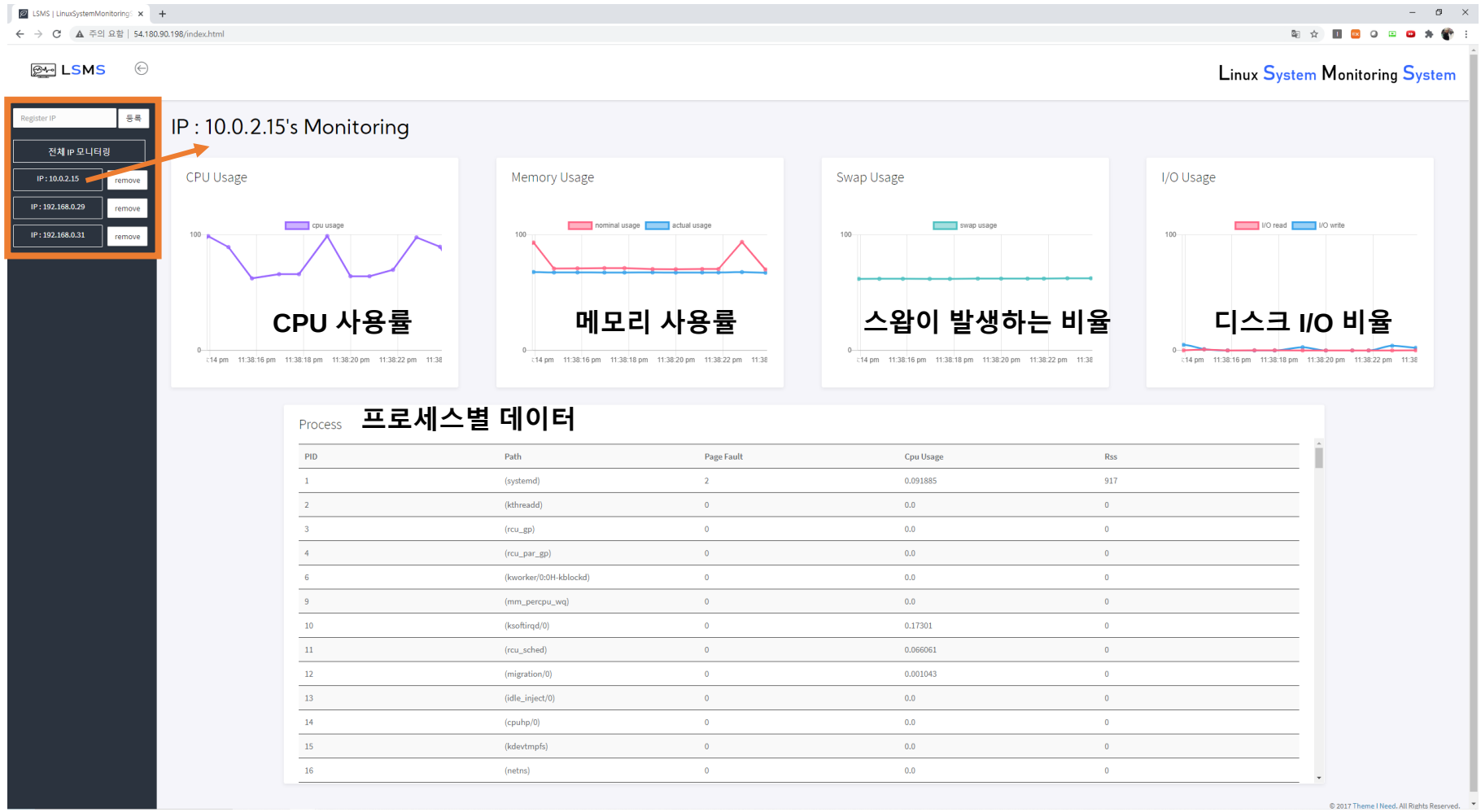
상호 배제가 일어나는 구간

```c
int main(){
        char *name[] = {"sejin", "runa"};
        pthread_t tid[2];
        pthread_attr_t attr[2];

        pthread_attr_init(&attr[0]);
        pthread_attr_init(&attr[1]);

        pthread_spin_init(&lock, PTHREAD_PROCESS_PRIVATE);

        pthread_create(&tid[0], &attr[0], worker, name[0]); //create thread
        pthread_create(&tid[1], &attr[1], worker, name[1]);

        pthread_join(tid[0], NULL);
        pthread_join(tid[1], NULL);

        printf("total sum = %d \n", sum);

        pthread_spin_destroy(&lock);

        return 0;
}
```

02. 리눅스 시스템 모니터링 시스템 동작 원리

02. 리눅스 시스템 모니터링 시스템 동작 원리

- 개발한 시스템 웹페이지



02. 리눅스 시스템 모니터링 시스템 동작 원리

- 리눅스에서 시스템 정보는 /proc 에서 볼 수 있음

```
sejin@sejin-virtual-machine:~$ cd /proc
sejin@sejin-virtual-machine:/proc$ ls
1     1186  132   17    228   256   4484  83   99           modules
10    119   1326  18    229   257   4485  830  acpi         mounts
100   1190  133   1982  23    258   473   833  buddyinfo    mpt
101   1195  1332  2     230   259   474   84   bus          mtrr
1015  12    1341  20    231   26    481   85   cgroups      net
102   120   1343  203   232   260   484   86   cmdline      pagetypeinfo
103   1201  1348  204   233   261   564   87   consoles     partitions
104   121   1349  205   234   262   6     878  cpuinfo      pressure
105   1211  1357  2056  235   263   657   88   crypto       sched_debug
106   122   1372  206   236   264   658   880  devices      schedstat
107   1227  1373  2068  2367  265   663   883  diskstats    scsi
108   1229  1376  207   237   27    664   887  dma          self
109   123   1380  208   2377  28    667   889  driver       slabinfo
1092  124   13906 209   238   29    669   892  execdomains  softirqs
11    125   14    21    239   293   675   896  fb           stat
110   126   1411  210   2396  294   676   9    filesystems  swaps
111   1265  143   211   24    295   693   90   fs           sys
1113  127   14448 213   240   297   696   902  interrupts   sysrq-trigger
1118  1271  1448  214   241   3     705   91   iomem        sysvipc
112   128   1458  215   2416  30    706   914  ioports      thread-self
1122  1284  146   216   242   31    708   921  irq          timer_list
113   1285  14648 217   243   326   711   93   kallsyms     tty
1130  1287  1479  218   244   327   720   938  kcore        uptime
114   1288  14993 219   245   366   767   94   key-users    version
1145  129   15    22    246   385   778   942  keys         version_signature
115   1293  15056 220   247   394   78    95   kmsg         vmallocinfo
116   1296  15103 221   248   396   786   956  kpagecgroup  vmstat
1162  1297  15131 222   249   397   79    96   kpagecount   zoneinfo
1168  13    15233 223   250   398   80    962  kpageflags
1169  130   15248 2238  251   399   801   97   loadavg
117   1303  15259 224   252   4     82    972  locks
1171  1304  15281 225   253   407   820   976  mdstat
1177  1307  159   226   254   409   821   98   meminfo
118   1319  16    227   255   425   828   984  misc
```

```
sejin@sejin-virtual-machine:/proc$ cat meminfo
MemTotal:        4002248 kB
MemFree:         1270388 kB
MemAvailable:    2639956 kB
Buffers:          113416 kB
Cached:          1402732 kB
SwapCached:            0 kB
Active:          1418672 kB
Inactive:         748884 kB
Active(anon):     652652 kB
Inactive(anon):     1912 kB
Active(file):     766020 kB
Inactive(file):   746972 kB
Unevictable:          16 kB
Mlocked:              16 kB
SwapTotal:       3998716 kB
SwapFree:        3998716 kB
Dirty:                24 kB
Writeback:             0 kB
AnonPages:        651464 kB
Mapped:           286192 kB
Shmem:              3160 kB
KReclaimable:     122916 kB
Slab:             237664 kB
SReclaimable:     122916 kB
SUnreclaim:       114748 kB
KernelStack:       10240 kB
PageTables:        12532 kB
NFS_Unstable:          0 kB
Bounce:                0 kB
WritebackTmp:          0 kB
CommitLimit:     5999840 kB
Committed_AS:    3557376 kB
VmallocTotal:   34359738367 kB
```

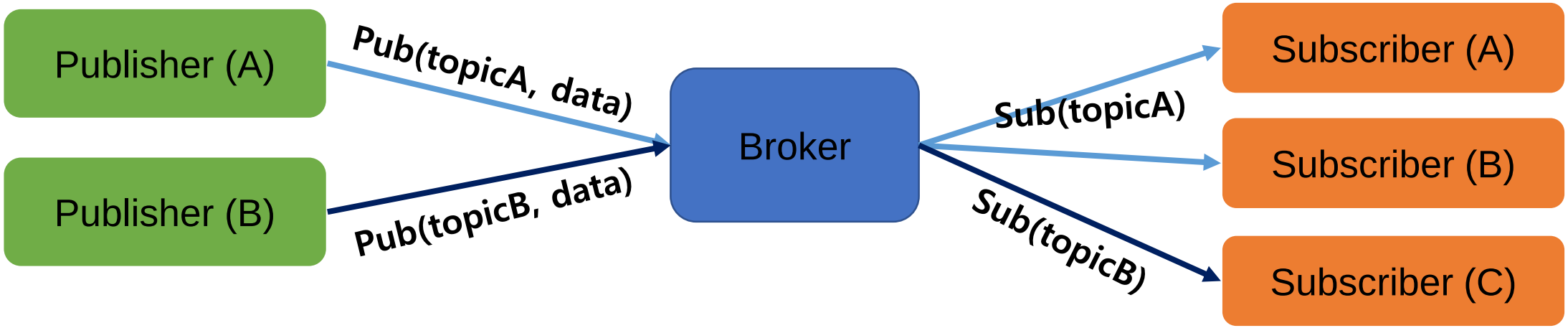
02. 리눅스 시스템 모니터링 시스템 동작 원리

- MQTT란?

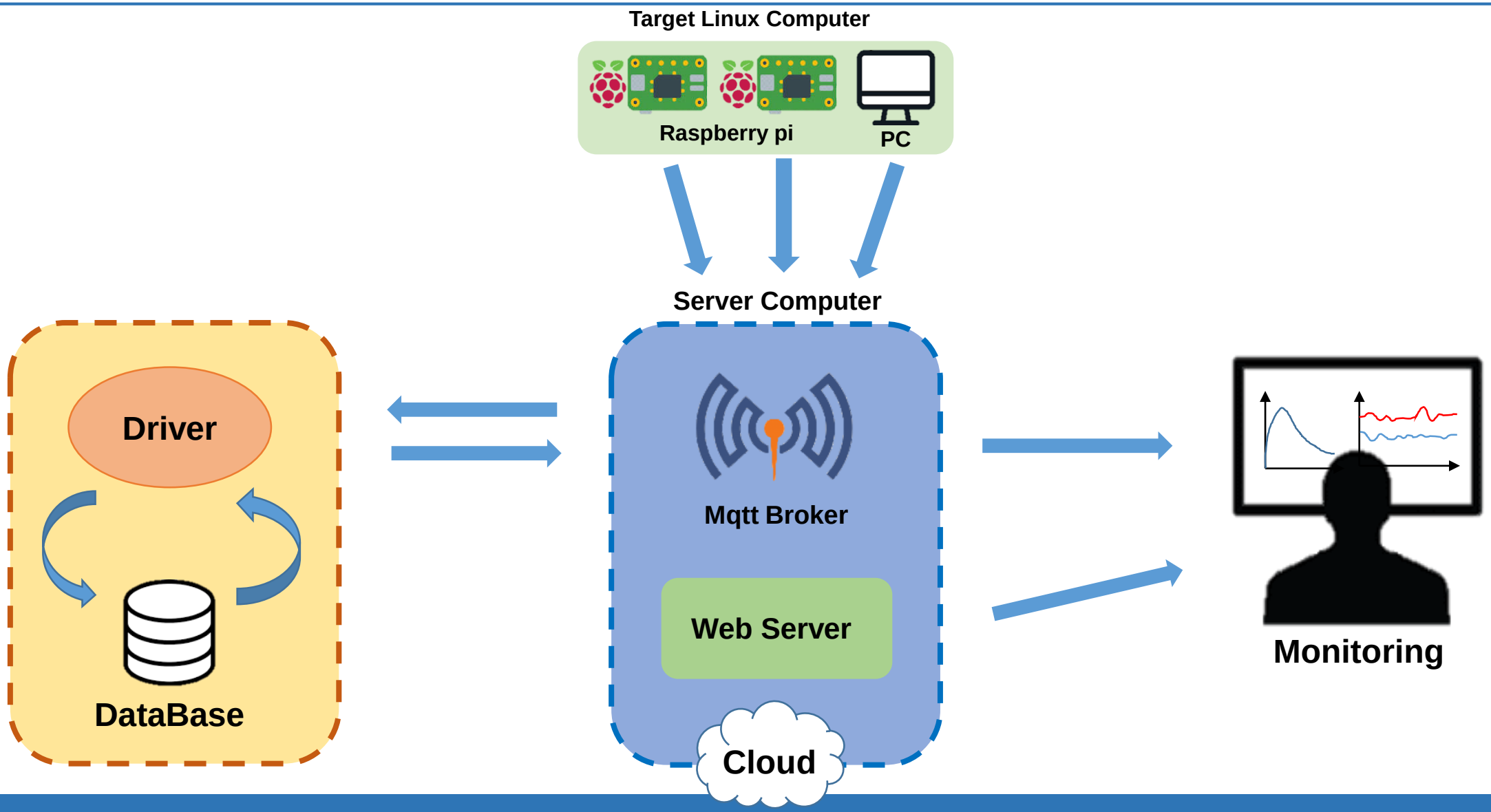
- HTTP, TCP 등의 통신처럼 클라이언트-서버 구조가 아닌,

  Broker, Publisher, Subscriber 구조로 이루어지는 통신 프로토콜

- 최소한의 전력과 패킷으로 통신하기 때문에 IoT와 모바일 어플리케이션 등의 통신에 적합



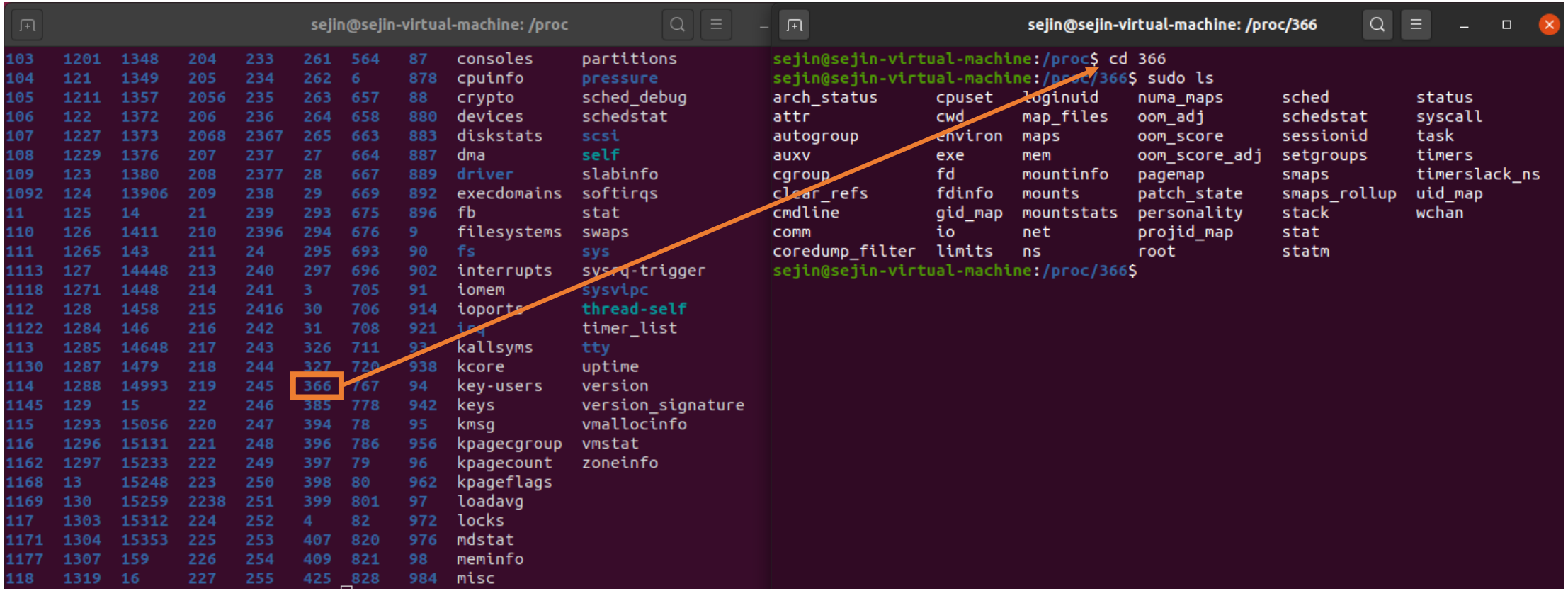
02. 리눅스 시스템 모니터링 시스템 동작 원리



03. 파일 시스템 코드 분석

03. 파일 시스템 코드 분석

- ps 명령어를 사용하지 않고 현재 실행 중인 프로세스의 pid 알아내기



03. 파일 시스템 코드 분석

- ps명령어를 사용하지 않고 현재 실행 중인 프로세스의 pid 알아내기

함수 정의

```c
#include <dirent.h>
//function to check if a struct dirent from /proc is a PID folder
int is_pid_folder(const struct dirent *entry){
        const char *p;

        for (p = entry->d_name; *p; p++){
                if(!isdigit(*p))
                        return 0;
        }

        return 1;
}
```

Main에서 함수 호출

```c
int main (void){
        FILE* pidFile;
        DIR *procDir;
        struct dirent *entry;
        char path[6 + 256 + 5]; // /proc/ + d_name + /stat

        while(1){

                //Open /proc directory
                procDir = opendir("/proc");
                if(!procDir){
                        perror("opendir failed");
                        return 1;
                }

                //Iterate through all files and folders of /proc
                while((entry = readdir(procDir))){
                        //Skip anything that is not a PID folder
                        if(!is_pid_folder(entry))
                                continue;

                        //Try to open /proc/[PID]/stat
                        snprintf(path, sizeof(path), "/proc/%s/stat", entry->d_name);
                        pidFile = fopen(path, "r");

                        if(!pidFile){
                                perror(path);
                                continue;
```

03. 파일 시스템 코드 분석

- swap.c

```
sejin@sejin-virtual-machine:/proc$ cat meminfo
MemTotal:        4002248 kB
MemFree:         1270388 kB
MemAvailable:    2639956 kB
Buffers:          113416 kB
Cached:          1402732 kB
SwapCached:            0 kB
Active:          1418672 kB
Inactive:         748884 kB
Active(anon):     652652 kB
Inactive(anon):     1912 kB
Active(file):     766020 kB
Inactive(file):   746972 kB
Unevictable:          16 kB
Mlocked:              16 kB
SwapTotal:       3998716 kB
SwapFree:        3998716 kB
Dirty:                24 kB
Writeback:             0 kB
AnonPages:        651464 kB
Mapped:           286192 kB
Shmem:              3160 kB
KReclaimable:     122916 kB
Slab:             237664 kB
SReclaimable:     122916 kB
SUnreclaim:       114748 kB
KernelStack:       10240 kB
PageTables:        12532 kB
NFS_Unstable:          0 kB
Bounce:                0 kB
WritebackTmp:          0 kB
CommitLimit:     5999840 kB
Committed_AS:    3557376 kB
VmallocTotal:   34359738367 kB
```

```c
#define ONE_LINE 80
#define PAST 0
#define PRESENT 1
#define SWAP_NUM 3
#define PORT 9001

enum swaps{TOTAL, FREE, USAGE} swap_enum;

int file_scan_certain(FILE* targetFile, char target[]){

        int result = 9999999;
        char temp[100];

        while(!feof(targetFile)){

                fscanf(targetFile, "%s", temp);
                if(!strcmp(temp, target)){
                        fscanf(targetFile, "%d", &result);
                        fflush(stdin);  return result;
                }
                else{
                        fscanf(targetFile, "%*d %*s");
                        fflush(stdin);
                }
        }       //end while

        return -1;
}
```

03. 파일 시스템 코드 분석

- swap.c

```c
int main (void){
        float swap[SWAP_NUM] = {0};
        char broker_address[100];
        char hostIP[40];
        char instruct[400] = {0};

        FILE* statFile;
        FILE* meminfoFile;

        //get broker IP address /in getBrokerIP.c
        getBrokerIP(broker_address);

        //get host IP   /in getIP.c
        getIP(hostIP);

        meminfoFile = fopen("/proc/meminfo", "r");
        swap[TOTAL] = file_scan_certain2(meminfoFile, "SwapTotal:"); //fixed space

        while(1){
                 meminfoFile = fopen("/proc/meminfo", "r"); //open meminfo
                swap[FREE]  = file_scan_certain2(meminfoFile, "SwapFree:");
                swap[USAGE] = (swap[TOTAL] - swap[FREE]) / swap[TOTAL];

                float swap_usage = 100*swap[USAGE];
               // sprintf(swap_value, "%f", swap[USAGE]);
               //printf("Swap usage : %f %%\n", swap_usage);

                sprintf(instruct, "sudo mosquitto_pub -t 'mon/storeDB/SWAP' -h %s -m '{ \"IP\" : \"%s\", \"timestamp\" : %d, \"swap_usage\" : %f }'",
                                                                broker_address, hostIP, (int)time(NULL), swap_usage);

                printf("%s\n", instruct);
                system(instruct);

                fclose(meminfoFile);

                sleep(1);
        }

        return 0;
}
```

Swap 비율 구하는 방법

$$100 \times \frac{\text{현재 사용 중인 swap 메모리 양}}{\text{swap에 할당된 전체 메모리의 크기}}$$

04. 동작 시연

감사합니다