

과제 1

-리눅스 커널 컴파일, newps구현-

학번: 20212211

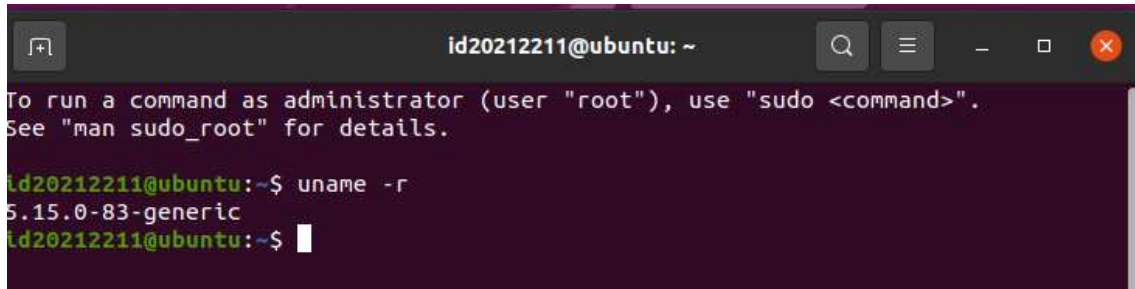
이름: 권대호

목차

1. 리눅스 커널 컴파일 및 설치	3
2. newps 명령어 구현	4

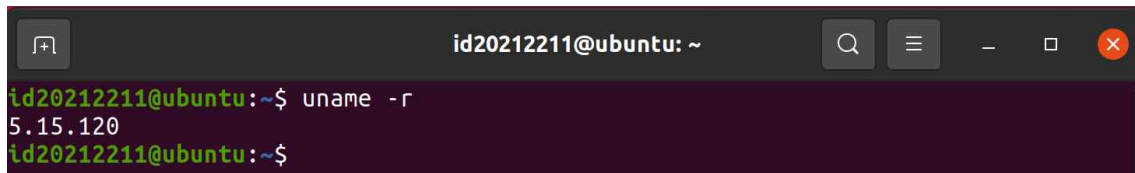
1. 리눅스 커널 컴파일 및 설치

가. 컴파일된 커널 설치 이전 화면



```
id20212211@ubuntu: ~  
To run a command as administrator (user "root"), use "sudo <command>".  
See "man sudo_root" for details.  
  
id20212211@ubuntu:~$ uname -r  
5.15.0-83-generic  
id20212211@ubuntu:~$
```

나. 컴파일된 커널 설치 이후 화면



```
id20212211@ubuntu: ~  
id20212211@ubuntu:~$ uname -r  
5.15.120  
id20212211@ubuntu:~$
```

2. newps 명령어 구현

가. 소스코드

```
#include <stdio.h>
#include <dirent.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>

int main() {
    DIR *proc;
    struct dirent *location;

    proc = opendir("/proc");
    if (proc == NULL) {
        perror("opendir");
        return 1;
    }

    long clk_tck = sysconf(_SC_CLK_TCK);
    printf("    PID TTY          TIME CMD\n");

    int self_tty_nr;
    {
        int pid = getpid();
        char path[10000];
        snprintf(path, sizeof(path), "/proc/%d/stat", pid);
        FILE *file_stat = fopen(path, "r");
        if (file_stat == NULL) {
            perror("fopen");
            return 1;
        }
        fscanf(file_stat, "%*d %s %c %d %d %d %d", &self_tty_nr);
        fclose(file_stat);
    }

    int tty_nr_values[1024] = {0};
    int tty_nr_count = 0;
```

```

DIR *proc2;
struct dirent *location2;

proc2 = opendir("/proc");
if (proc2 == NULL) {
    perror("opendir");
    return 1;
}

while ((location2 = readdir(proc2)) != NULL) {
    char path[10000];
    int pid, tty_nr;

    if (sscanf(location2->d_name, "%d", &pid) != 1) {
        continue;
    }

    snprintf(path, sizeof(path), "/proc/%d/stat", pid);
    FILE *file_stat = fopen(path, "r");

    if (file_stat == NULL) {
        perror("fopen");
        return 1;
    }

    fscanf(file_stat, "%*d %*s %*c %*d %*d %*d %d", &tty_nr);

    fclose(file_stat);

    int found = 0;
    for (int i = 0; i < tty_nr_count; i++) {
        if (tty_nr_values[i] == tty_nr) {
            found = 1;
            break;
        }
    }

    if (!found && tty_nr_count <
sizeof(tty_nr_values)/sizeof(tty_nr_values[0])) {
        tty_nr_values[tty_nr_count++] = tty_nr;
    }
}

```

```

    }

    }
    closedir(proc2);

    while ((location = readdir(proc)) != NULL) {
        char path[10000];
        char cmdline[1000];
        int pid, tty_nr;
        unsigned long usermode_time, kernelmode_time;

        if (sscanf(location->d_name, "%d", &pid) != 1) {
            continue;
        }
        snprintf(path, sizeof(path), "/proc/%d/cmdline", pid);
        FILE *file_cmdline = fopen(path, "r");

        snprintf(path, sizeof(path), "/proc/%d/stat", pid);
        FILE *file_stat = fopen(path, "r");

        if (file_cmdline == NULL || file_stat == NULL) {
            perror("fopen");
            return 1;
        }

        fgets(cmdline, sizeof(cmdline), file_cmdline);
        fscanf(file_stat, "%*d %*s %*c %*d %*d %*d %*d %*d %*u %*u %*u %*u %*u %*u %*u %*u", &tty_nr, &usermode_time, &kernelmode_time);

        if (tty_nr != self_tty_nr) {
            continue;
        }

        double seconds = (double)(usermode_time + kernelmode_time) /
clk_tck;
        char *cmdline_skip = cmdline;
        if (cmdline[0] == '.' && cmdline[1] == '/') {
            cmdline_skip += 2;
        }
        printf("%7d pts/%-5d%02ld:%02ld:%02ld %s\n", pid, tty_nr_count-3,

```

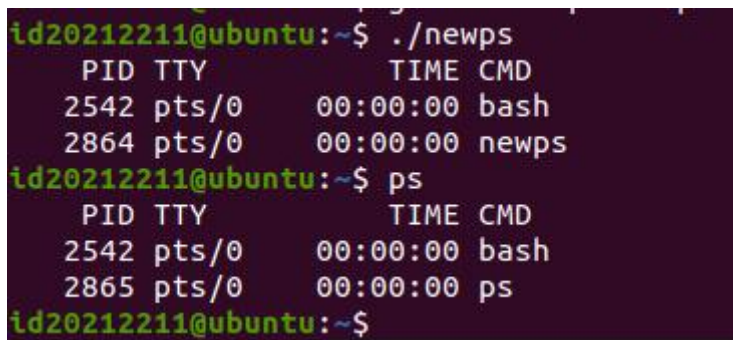
```

(long)(seconds / 3600), (long)(seconds / 60) % 60, (long)seconds % 60,
cmdline_skip);

    fclose(file_cmdline);
    fclose(file_stat);
}
closedir(proc);
return 0;
}

```

나. 실행결과



```

id20212211@ubuntu:~$ ./newps
  PID TTY          TIME CMD
 2542 pts/0        00:00:00 bash
 2864 pts/0        00:00:00 newps
id20212211@ubuntu:~$ ps
  PID TTY          TIME CMD
 2542 pts/0        00:00:00 bash
 2865 pts/0        00:00:00 ps
id20212211@ubuntu:~$

```

다. 코드설명

newps.c는 PID, TTY, TIME, CMD를 출력한다.

이 프로그램의 실행과정을 간단히 설명하면,

- 1) /proc 디렉토리를 연다.
- 2) proc2 디렉토리를 내부에서 다시 생성한 다음 pid값으로 이루어진 디렉토리를 검색한다.
 - 2-1) stat정보를 가져온 다음 pid값이 순차적으로 검색될때마다 tty_nr값 구한다.
 - 2-2) 구한 tty_nr값이 tty_nr_value배열있는지 확인하고 없다면 추가한다.
 - 2-3) tty_nr_value의 크기를 tty_nr_count에 저장한다. 모든 작업이 끝나면 proc2를 닫는다.
- 3) pid값으로 이루어진 디렉토리를 검색한다. 더 이상 검색이 안되면 프로그램을 종료한다.
 - 3-1) 검색된 pid값으로 디렉토리를 연다음 stat정보와 cmdline정보를 가져온다.
 - 3-2) stat정보에서 usermode시간(14번), kernelmode시간(15번), 이 3가지 정보를 가져온다.
 - 3-3) utime과 stime으로 시간을 구한다.
 - 3-4) 현재 명령어가 실행되는 tty_nr값을 self_tty_nr에 저장하고, self_tty_nr과 tty_nr값이 같으면 출력대상으로 선정한다.
 - 3-5) './'으로 시작하는 cmdline값은 ./를 떼고 출력한다. 이는 cmdline_skip이라는 배열을 만들고 ./로 시작하는 것이 확인되면 그 이후부터 cmdline_skip에 넣고, 그렇지 않다면 cmdline_skip을 cmdline과 동일하게 설정한다.
 - 3-6) pid, tty, time, cmd를 출력한다. 여기서 tty값은 tty_nr_count값-3이다.

pid는 /proc에서 숫자로 이루어진 디렉토리 이름을 통해 pid값으로 구한다. tty, time, cmd값은 /proc/[pid]/stat에서 tty, time정보, /proc/[pid]/cmdline에서 cmd정보를 가져오기 위해 각각 FILE file_stat;, FILE file_cmdline;으로 가져온다.

time값은 /stat에서 utime과 stime을 구해 두 값을 더한 값을 초당 틱수로 나눠서 구한다. 초당 틱수는 sysconf(_SC_CLK_TCK)을 통해 구한다.

cmd는 cmdline 그대로 출력한다.