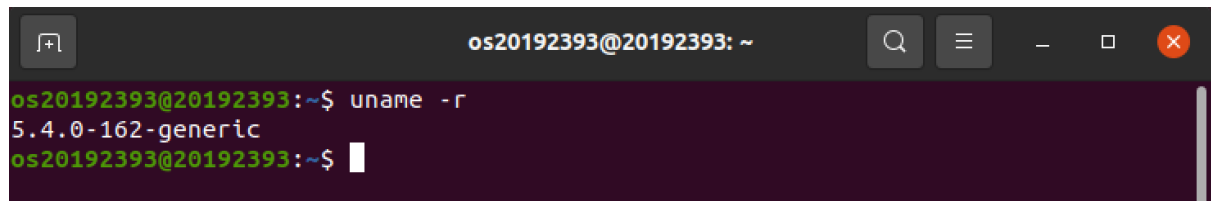




운영체제

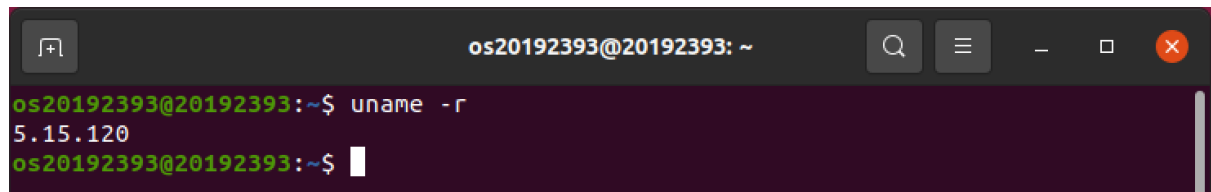
과목명	운영체제
교수명	김철홍
학 과	컴퓨터학부
학 번	20192393
이 름	김현우
제출일	2023.09.18

1. 컴파일된 커널 설치 이전 화면



```
os20192393@20192393: ~  
os20192393@20192393:~$ uname -r  
5.4.0-162-generic  
os20192393@20192393:~$
```

2. 컴파일된 커널 설치 이후 화면



```
os20192393@20192393: ~  
os20192393@20192393:~$ uname -r  
5.15.120  
os20192393@20192393:~$
```

3. newps 구현 코드 설명

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <dirent.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <stdbool.h>
#include <errno.h>
#include <ctype.h>
#include <fcntl.h>

#define BUFFER_SIZE 1024
#define PATH_LEN 1024
#define TTY_LEN 32

pid_t myPid;
uid_t myUid;
char myPath[PATH_LEN];
char myTTY[TTY_LEN];

char cmdline[BUFFER_SIZE];
unsigned long time;
char cpuTime[256];

void getTTY(char path[PATH_LEN], char tty[TTY_LEN]); //path 에 대한 tty 얻는 함수
void getProcessCMDandCPUtime(char* string); // PID 에 대한 CMD 와 cpu 사용시간을
구하는 함수
uid_t getUID(char path[PATH_LEN]); // UID 를 얻는 함수 추가

int main() {

    // 헤더 출력
    printf("PID\tTTY\tTIME\tCMD\n");

    // /proc 디렉토리 열기
    DIR *dir;
    if((dir = opendir("/proc")) == NULL){
        fprintf(stderr, "error for /proc\n");
        exit(1);
    }

    myPid = getpid(); //자기 자신의 pid
    myUid = getUID("/proc/self/status"); //자기 자신의 uid

    char pidPath[PATH_LEN];
    memset(pidPath, '\0', PATH_LEN);
    sprintf(pidPath, "%d", myPid);
```

```

strcpy(myPath, "/proc");           //자기 자신의 /proc 경로 획득
strcat(myPath, pidPath);

getTTY(myPath, myTTY);             //자기 자신의 tty 획득
for(int i = strlen("pts/"); i < strlen(myTTY); i++){
    if(!isdigit(myTTY[i])){
        myTTY[i] = '\0';
        break;
    }
}

struct dirent *dentry;
while((dentry = readdir(dir)) != NULL){ // /proc 디렉터리 내 하위 파일들 탐색 시작

    char path[PATH_LEN];           //디렉터리의 절대 경로 저장
    memset(path, '\0', PATH_LEN);
    strcpy(path, "/proc");
    strcat(path, "/");
    strcat(path, dentry->d_name);

    struct stat statbuf;
    if(stat(path, &statbuf) < 0){ //디렉터리의 stat 획득
        fprintf(stderr, "stat error for %s\n", path);
        exit(1);
    }

    if(!S_ISDIR(statbuf.st_mode)) //디렉터리가 아닐 경우 skip
        continue;

    int len = strlen(dentry->d_name);
    bool isPid = true;
    for(int i = 0; i < len; i++){ //디렉터리가 PID 인지 찾기
        if(!isdigit(dentry->d_name[i])){ //디렉터리명 중 숫자 아닌 문자가 있을
            isPid = false;
            break;
        }
    }

    char tty[TTY_LEN];
    if(!isPid)
        continue;
    else{
        memset(tty, '\0', TTY_LEN);
        getTTY(path, tty); //TTY 획득
        if(strcmp(tty, myTTY)) //자기 자신과 tty 다를 경우
            continue;
    }
}

```

경우

```

        uid_t processUid = getUID(strcat(path, "/status")); // /proc/pid/status
        경로에서 uid 값 구하기
        if (processUid != myUid)
            continue;

        getProcessCMDandCPUTime(dentry->d_name);

        printf("%s\t%s\t%s\t%s\n", dentry->d_name, tty, cpuTime, cmdline + 1);
    }
    closedir(dir);
    return 0;
}

//path 에 대한 tty 얻는 함수
void getTTY(char path[PATH_LEN], char tty[TTY_LEN]) {
    char fdZeroPath[PATH_LEN]; //0 번 fd 에 대한 절대 경로
    memset(tty, '\0', TTY_LEN);
    memset(fdZeroPath, '\0', TTY_LEN);
    strcpy(fdZeroPath, path);
    strcat(fdZeroPath, "/fd/0");

    if(access(fdZeroPath, F_OK) < 0){ //fd 0 이 없을 경우

        char statPath[PATH_LEN]; // /proc/pid/stat 에 대한 절대 경로
        memset(statPath, '\0', PATH_LEN);
        strcpy(statPath, path);
        strcat(statPath, "/stat");

        FILE *statFp;
        if((statFp = fopen(statPath, "r")) == NULL){ // /proc/pid/stat open
            fprintf(stderr, "fopen error %s %s\n", strerror(errno), statPath);
            sleep(1);
            return;
        }

        char buf[BUFFER_SIZE];
        for(int i = 0; i <= 6; i++){ // 7 행까지 read 해 TTY_NR 획득
            memset(buf, '\0', BUFFER_SIZE);
            fscanf(statFp, "%s", buf);
        }
        fclose(statFp);

        int ttyNr = atoi(buf); //ttyNr 정수 값으로 저장

        DIR *dp;
        struct dirent *dentry;
        if((dp = opendir("/dev")) == NULL){ // 터미널 찾기 위해 /dev 디렉터리 open
            fprintf(stderr, "opendir error for %s\n", "/dev");
            exit(1);
        }
    }
}

```

```

char nowPath[PATH_LEN];

while((dentry = readdir(dp)) != NULL){ // /dev 디렉터리 탐색
    memset(nowPath, '\0', PATH_LEN); // 현재 탐색 중인 파일 절대 경로
    strcpy(nowPath, "/dev");
    strcat(nowPath, "/");
    strcat(nowPath, dentry->d_name);

    struct stat statbuf;
    if(stat(nowPath, &statbuf) < 0){ // stat 획득
        fprintf(stderr, "stat error for %s\n", nowPath);
        exit(1);
    }
    if(!S_ISCHR(statbuf.st_mode)) //문자 디바이스 파일이 아닌 경우 skip
        continue;
    else if(statbuf.st_rdev == ttyNr){ //문자 디바이스 파일의 디바이스 ID 가
ttyNr 과 같은 경우
        strcpy(tty, dentry->d_name); //tty 에 현재 파일명 복사
        break;
    }
}
closedir(dp);

if(!strlen(tty)) // /dev 에서도 찾지 못한 경우
    strcpy(tty, "?"); //nonTerminal
}
else{
    char symLinkName[BUFFER_SIZE];
    memset(symLinkName, '\0', BUFFER_SIZE);
    if(readlink(fdZeroPath, symLinkName, BUFFER_SIZE) < 0){
        fprintf(stderr, "readlink error for %s\n", fdZeroPath);
        exit(1);
    }
    if(!strcmp(symLinkName, "/dev/null")) //symbolic link 로 가리키는 파일이
/dev/null 일 경우
        strcpy(tty, "?"); //nonTerminal
    else
        sscanf(symLinkName, "/dev/%s", tty); //그 외의 경우 tty 획득
}
return;
}

void getProcessCMDandCPUTime(char* string) {
    char path[64];
    char line[BUFFER_SIZE];
    unsigned long time, stime;
    int i;
    char flag;

```

```

// PID를 이용하여 stat 파일 경로 생성
sprintf(path, "/proc/%s/stat", string);

// stat 파일 열기
FILE *file = fopen(path, "r");
if (file == NULL) {
    perror("Error opening stat file");
}

fscanf(file, "%d%s%c%c%c", &i, cmdline, &flag, &flag, &flag);
cmdline[strlen(cmdline) - 1] = '\0'; // CMD 구하기

// utime(14 번째 필드)과 stime(15 번째 필드) 값 추출
for (int i = 0; i < 11; i++)
    fscanf(file, "%lu", &time);
    fscanf(file, "%lu", &stime);

time = (int)((double)(time + stime) / sysconf(_SC_CLK_TCK)); // utime + stime
으로 CPUtime 구하기
sprintf(cpuTime, "%02lu:%02lu:%02lu", (time / 3600) % 3600, (time / 60) % 60,
time % 60);

// 파일 닫기
fclose(file);
}

// UID를 얻는 함수
uid_t getUID(char path[PATH_LEN]) {
    FILE* statusFile = fopen(path, "r");
    if (statusFile == NULL) {
        perror("Error opening status file");
        exit(1);
    }

    char line[BUFFER_SIZE];
    char tag[64];
    uid_t uid = 0;

    while (fgets(line, sizeof(line), statusFile) != NULL) {
        if (sscanf(line, "%63s %u", tag, &uid) == 2) {
            if (strcmp(tag, "Uid:") == 0) {
                break;
            }
        }
    }

    fclose(statusFile);
    return uid;
}

```

4. newps 실행 결과 캡처

```
os20192393@20192393:~$ gcc newps.c -o newps
os20192393@20192393:~$ ./newps
PID      TTY      TIME    CMD
2010     pts/0    00:00:00      bash
4689     pts/0    00:00:00      ./newps
os20192393@20192393:~$ ps
  PID TTY          TIME CMD
  2010 pts/0    00:00:00 bash
  4690 pts/0    00:00:00 ps
```