7 Team Project Report
<Implementation and Management of miniOS TOP Command>

Managing miniOS Top Command

07. Team

김민석(20202894),
김영민(20201576),
안지수(20221602),
박시운(20221597)

Our team’s project index

01

1.프로젝트 목적과 배경, 기능

-What is a top command?
-Project Overview

02

2.코드 구조와 주요 알고리즘 설명

- Describe the roles and functions of each component

03

3.추가한 성능과 기능

- 직접 구현한 top을 위해 접근한 기능

04

4. 결론

- 프로젝트 결과 요약

What is a top command?

"시스템의 현재 상태를 실시간으로 모니터링하는 도구"

✅ **Key Features:**

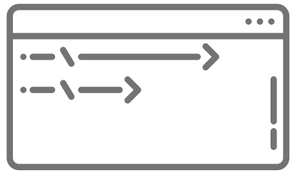
- CPU, 메모리 사용률 표시
- 실행 중인 프로세스 목록 제공
- 프로세스별 상세 정보 표시
  (PID, 사용자, CPU/메모리 사용량 등)

✅ **How to use:**

`top` 명령어 입력 후,
원하는 정보를 실시간으로 확인

Project Overview

top command
반환되는 정보 확인



command를 활용하여
실제 프로세스 추출

구현 및 출력
인터페이스 설계

Phase 1

<정보 수집 및 분석>
`strace` 명령어로 시스템 호출 분석

Phase 2

<System Call 분석>
`strace -o` 옵션으로 결과 저장 및 분석

Phase 3

<확인>
cat 명령어를 통해 표준 출력으로 읽을 수 있는
파일은 터미널에서 직접 확인

Phase 4

<구현 시작>
기존 `top` 명령어 기능 중 구현할 내용 선별 후
팀원 간 분담

Phase 5

<인터페이스 구현>
ncurses 라이브러리 사용: 터미널 내 인터페이스 구현

Describe the roles and functions of each component

1 ROW: System Information Row

“시스템의 현재 시간, 업타임, 로그인한 사용자 수, 평균 부하(load average)를 표시“

```
top - 22:16:23 up 13:30, load average: 0.00, 0.00, 0.00
Tasks:      6 total,      0 running,     0 uninterruptible_slee
Kib Mem :   7981560 total,    7803504 free,     107516 used,
%Cpu(s):    0.0 us,   0.0 sy,   0.0 ni, 801.0 id,   0.0 wa,   0.0


PID   PR   NI    VIRT    RES   SHR   S   %CPU    MEM   TIME+   COMMAND
  1   20    0    1832   1108     0       0.0    0.0   0:00   init
 13   20    0    2172    372     0       0.0   36.3   0:00   init
 14   20    0    2180    372     0       0.0   36.3   0:00   init
 15   20    0    6168   5144     0       0.0    0.0   0:00   bash
207   20    0    7796   3356     0       0.0    0.0   0:00   top
221   20    0   14180   3676     0       0.0    0.0   0:00   minios
```

```c
/*1. Uptime 가져오기*/
    time_t uptime;
    uptime = get_uptime();          //os 부팅 후 지난 시각
    char buf[BUFFER_SIZE];
    /*****  1행 UPTIME 출력    *****/
    /*2. 현재 시각 문자열 생성*/
    char nowStr[128] = { 0 };       // 현재 시각 문자열을 초기화
    time_t now = time(NULL);      // 현재 시각을 얻기
    struct tm* tmNow = localtime(&now);  // 현재 시각을 struct tm으로 변환

    // 현재 시각을 "top - HH:MM:SS " 형식으로 nowStr에 저장
    strftime(nowStr, sizeof(nowStr), "top - %H:%M:%S ", tmNow);

    /*3. Uptime 문자열 생성*/
    struct tm* tmUptime = localtime(&uptime);

    char upStr[128] = { 0 };  // uptime 문자열 초기화
    if (uptime < 60 * 60) {
        snprintf(upStr, sizeof(upStr), "%2d min", tmUptime->tm_min);
    }
    else if (uptime < 60 * 60 * 24) {
        snprintf(upStr, sizeof(upStr), "%2d:%02d", tmUptime->tm_hour, tmUptime->tm_min);
    }
    else {
        snprintf(upStr, sizeof(upStr), "%3d days, %02d:%02d", tmUptime->tm_yday, tmUptime->tm_hour,
tmUptime->tm_min);
    }

    /* 4. Load Average 가져오기 */
    FILE* loadAvgFp;
    long double loadAvg[3];
    if ((loadAvgFp = fopen(LOADAVG, "r")) == NULL) {
        fprintf(stderr, "fopen error for %s\n", LOADAVG);
        exit(1);
    }

    if (fgets(buf, BUFFER_SIZE, loadAvgFp) != NULL) {
        sscanf(buf, "%Lf%Lf%Lf", &loadAvg[0], &loadAvg[1], &loadAvg[2]);
    }
    fclose(loadAvgFp);


    /*5. 출력*/
    mvprintw(TOP_ROW, 0, "%sup %s, load average: %4.2Lf, %4.2Lf, %4.2Lf", nowStr, upStr, loadAvg[0],
loadAvg[1], loadAvg[2]);
```

```c
unsigned int total = 0, running = 0, sleeping = 0, stopped = 0, zombie = 0, uninterruptible_sleep = 0,
tracedORstopped = 0;
    total = procCnt;
    for (int i = 0; i < procCnt; i++) {
        if (!strcmp(procList[i].stat, "R")) //실행 중인 프로세스
            running++;
        else if (!strcmp(procList[i].stat, "D")) //불가피하게 대기 중인 프로세스
            uninterruptible_sleep++;
        else if (!strcmp(procList[i].stat, "S")) //대기 중인 프로세스
            sleeping++;
        else if (!strcmp(procList[i].stat, "T")) //정지된 프로세스
            stopped++;
        else if (!strcmp(procList[i].stat, "t")) //추적 중인 프로세스 또는 멈춤 상태인 프로세스
            tracedORstopped++;
        else if (!strcmp(procList[i].stat, "Z")) //좀비 상태인 프로세스
            zombie++;
    }

    mvprintw(TASK_ROW, 0, "Tasks:  %4u total,  %4u running, %4u uninterruptible_sleep, %4u sleeping,  %4u
stopped, %4u tracedORstopped, %4u zombie", total, running, uninterruptible_sleep, sleeping, stopped,
tracedORstopped, zombie);



    hertz = (unsigned int)sysconf(_SC_CLK_TCK); //os의 hertz값 얻기(초당 context switching 횟수)
    char buffer[BUFFER_SIZE]; // 0행
    uptime = get_uptime();
```

2 ROW: Task information row

"전체 작업 수, 실행 중인 작업 수, 유휴 상태의 작업 수, 중단된 작업 수, 좀비 작업 수를 표시합니다."

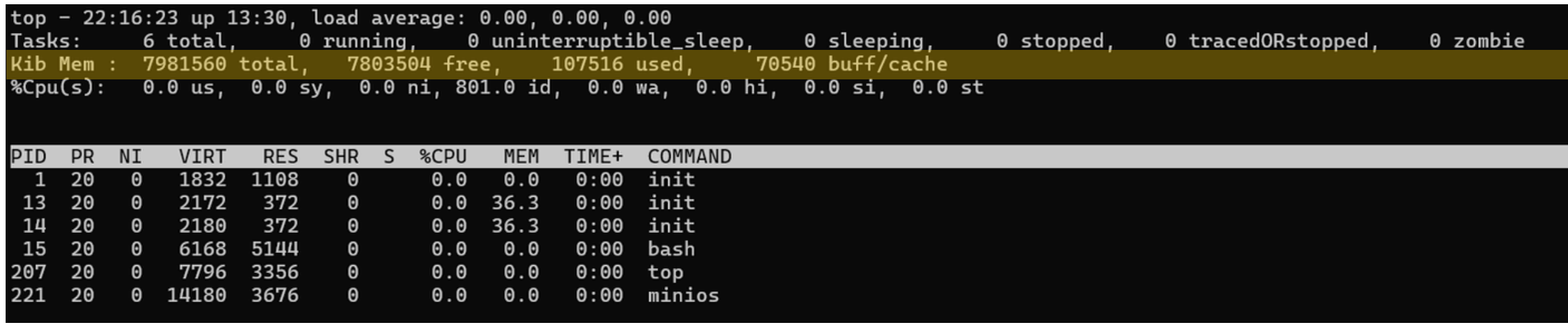
```
top - 22:16:23 up 13:30, load average: 0.00, 0.00, 0.00
Tasks:     6 total,     0 running,     0 uninterruptible_sleep,     0 sleeping,     0 stopped,     0 tracedORstopped,     0 zombie
Kib Mem :  7981560 total,   7803504 free,    107516 used,     70540 buff/cache
%Cpu(s):   0.0 us,  0.0 sy,  0.0 ni, 801.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st


PID  PR  NI    VIRT    RES  SHR  S  %CPU   MEM  TIME+  COMMAND
  1  20   0    1832   1108    0      0.0   0.0   0:00  init
 13  20   0    2172    372    0      0.0  36.3   0:00  init
 14  20   0    2180    372    0      0.0  36.3   0:00  init
 15  20   0    6168   5144    0      0.0   0.0   0:00  bash
207  20   0    7796   3356    0      0.0   0.0   0:00  top
221  20   0   14180   3676    0      0.0   0.0   0:00  minios
```

3 ROW: Physical memory information row

```
memUsed = memTotal - memFree - buffers - cached;
mvprintw(MEM_ROW, 0, "Kib Mem : %8lu total,  %8lu free,  %8lu used,  %8lu buff/cache"
         , memTotal, memFree, memUsed, buffers + cached);
fclose(meminfoFP);
```

“총 메모리, 사용 중인 메모리, 여유 메모리, 버퍼/캐시 메모리 양을 표시합니다.”



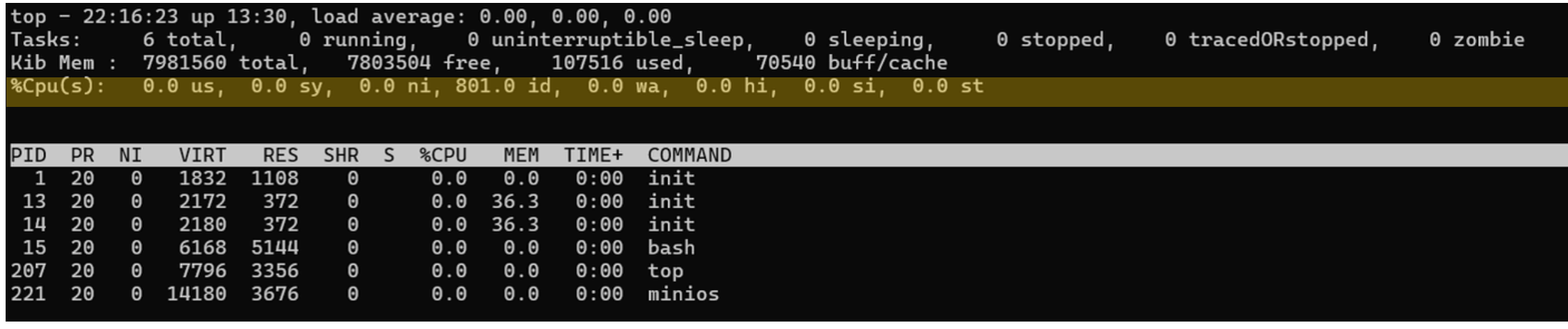
```
unsigned long nowTicks = 0;
long double results[CPUTicks] = { 0.0, };

if (beforeUptime == 0) {
    nowTicks = uptime * hertz;
    for (int i = 0; i < CPUTicks; i++) {
        results[i] = ticks[i];
    }
}
else {
    nowTicks = (uptime - beforeUptime) * hertz;
    for (int i = 0; i < CPUTicks; i++) {
        results[i] = ticks[i] - beforeTicks[i];
    }
}

for (int i = 0; i < CPUTicks; i++) {
    results[i] = (results[i] / nowTicks) * 100;
    if (isnan(results[i]) || isinf(results[i])) {
        results[i] = 0;
    }
}
```

4 ROW: CPU Status Row

“CPU 사용률을 여러
카테고리로 나누어 표시합니다.”



```
  int columnWidth[COLUMN_CNT] = {                    //column의 x축 길이 저장하는 배열
        strlen(PID_STR), strlen(PR_STR), strlen(NI_STR),
        strlen(VIRT_STR), strlen(RES_STR), strlen(SHR_STR), strlen(S_STR),
        strlen(CPU_STR), strlen(MEM_STR), strlen(TIME_P_STR), strlen(COMMAND_STR) };


/*--------------------- PID,PR,NU 등 최대 길이를 저장하는 코드 ----------------------------

    if(col >= COLUMN_CNT - 1){                     //COMMAND COLUMN만 출력하는 경우 (우측 화살
        startCol = COMMAND_IDX;                    //col: 사용자가 선택한 열의 인덱스
        endCol = COLUMN_CNT;
        maxCmd = COLS;                             //COMMAND 터미널 너비만큼 출력 가능, COLS:
창의 가로 길이
    }
    else{
        int i;
        for(i = col + 1; i < COLUMN_CNT; i++){
            startX[i] = columnWidth[i-1] + 2 + startX[i-1];
            if(startX[i] >= COLS){                 //COLUMN의 시작이 이미 터미널 너비 초과한 경우
                endCol = i;
                break;
            }
        }
        startCol = col;
        if(i == COLUMN_CNT){
            endCol = COLUMN_CNT;                       //COLUMN 전부 출력하는 경우
            maxCmd = COLS - startX[COMMAND_IDX];       //COMMAND 최대 출력 길이: COMMAND 터미
시작 x좌표
        }
    }
```

5 ROW: Process management row

"각 프로세스 정보에 대한 열(column) 헤더를 표시합니다. 각 열은 프로세스의 세부 정보를 나타냅니다"

```
top - 22:16:23 up 13:30, load average: 0.00, 0.00, 0.00
Tasks:     6 total,     0 running,     0 uninterruptible_slee
Kib Mem :  7981560 total,   7803504 free,    107516 used,
%Cpu(s):   0.0 us,  0.0 sy,  0.0 ni, 801.0 id,  0.0 wa,  0.0


PID  PR  NI    VIRT    RES  SHR  S  %CPU   MEM  TIME+  COMMAND
  1  20   0    1832   1108    0      0.0   0.0  0:00  init
 13  20   0    2172    372    0      0.0  36.3  0:00  init
 14  20   0    2180    372    0      0.0  36.3  0:00  init
 15  20   0    6168   5144    0      0.0   0.0  0:00  bash
207  20   0    7796   3356    0      0.0   0.0  0:00  top
221  20   0   14180   3676    0      0.0   0.0  0:00  minios
```

```
int columnWidth[COLUMN_CNT] = {
    strlen(PID_STR), strlen(PR_STR), strlen(NI_STR),
    strlen(VIRT_STR), strlen(RES_STR), strlen(SHR_STR), strlen(S_STR),
    strlen(CPU_STR), strlen(MEM_STR), strlen(TIME_P_STR),
    strlen(COMMAND_STR)
};

/* -----------PID,PR,NI 각각의 길이 차를 구해주는 코드--------------- */

else
{
    int i;
    for(i = col + 1; i < COLUMN_CNT; i++){
        startX[i] = columnWidth[i-1] + 2 + startX[i-1];
        if(startX[i] >= COLS){ /* COLUMN의 시작이 이미 터미널 너비 초과한 경우
            endCol = i;
            break;
        }
    }
    startCol = col;
    if(i == COLUMN_CNT){
        endCol = COLUMN_CNT; /* COLUMN 전부 출력하는 경우 */
        maxCmd = COLS - startX[COMMAND_IDX]; /* COMMAND 최대 출력 길이 */
    }
}
```

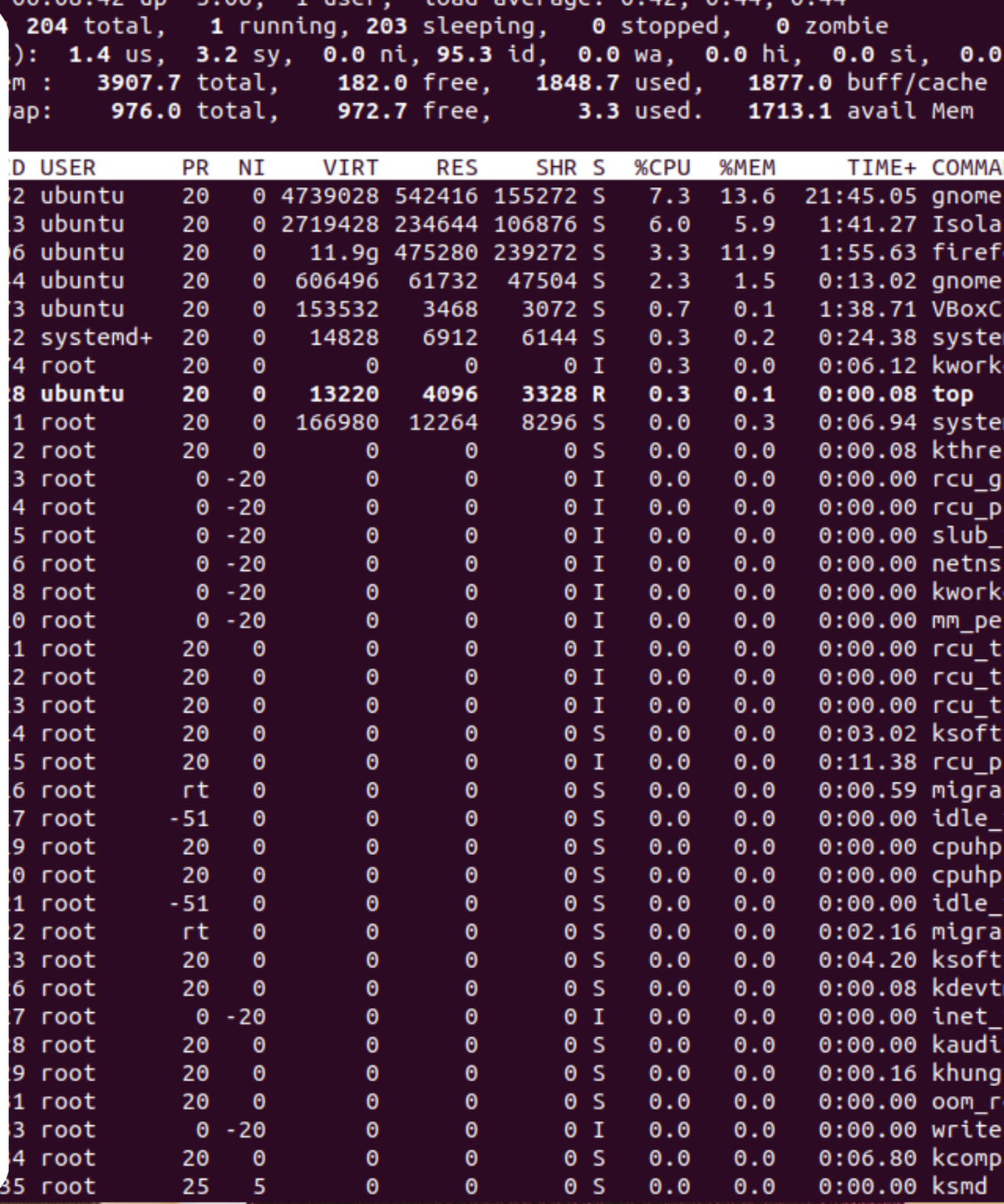
6 ROW : main

각 프로세스마다 칼럼 지정 후
동적으로 할당
(초기폭을 설정)

각 열의 시작위치와 출력 범위를 지정
(5행의 프로세스 관리 포맷을 계산하여
동일하게 해야할 필요가 있음

TOP helper

전반적인 top 구현을 위한
함수와 변수들을 모아 놓은 곳입니다.
실제 코드를 통해 자세한 설명을 하겠습니다.



1

top command ++

top command를 통해
효율적인 접근 및 관리를 할 수 없을까?

2

특정 정보만 볼 수는 없을까?

cpu scheduling도 보고 싶어!

Additional features

Implementation and Management of miniOS TOP Command

Thank You For Listening

07. Team

Do you have any questions for me?

김민석(20242894),
김영민(20241234),
안지수(20221602),
박시운(20221597)