운영체제

과목명	운영체제
교수명	김철홍
학 과	컴퓨터학부
학 번	20192393
이 름	김현우
제출일	2023.11.06

1 . 가산점 기능 구현 여부 (구현 성공)

2 . 가산점 기능 제출하는 소스코드 파일 리스트

- core.c : /usr/src/linux/linux-5.15.120/kernel/sched
- exit.c : /usr/src/linux/linux-5.15.120/kernel

3. 기본과제 테스트 프로그램 수행 시 먼저 실행해야하는 명령어

- gcc scheduler.c -lrt

4. 작업 설명

리눅스 커널의 스케줄링 정책을 변경하면서 생성되는 프로세스들의 실행 순서 및 실행 시간을 확인할 수 있는 프로그램이다.

스케줄링 정책은 CFS_DEFAULT, CFS_NICE, RT_FIFO, RT_RR 4 개의 스케줄링 정책을 적용한다.

scheduler.c

```c
#define _GNU_SOURCE
#include <sched.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <signal.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <sys/time.h>
#include <sys/mman.h>
#include <sys/resource.h>
#include <string.h>
#include <time.h>
#include <errno.h>
#include <signal.h>
#include <dirent.h>
#include <sys/wait.h>
#include <errno.h>

char printElapsedTime[256];
double averageElapsedTime = 0.0;

int status;
int pid[21];

int shm_fd;
char *shm_ptr;
const char *shm_name = "monitor_memory";

void print_menu(); // menu 출력하는 함수
void calculate(); // process 가 실행할 계산하는 함수
char* getElapsedTimeSpec(struct timespec Tstart, struct timespec Tend); // elpased time 구하는 함수
void addElapsedTime(const char *printElapsedTime); // elapsed time shared memory 에 저장하는 함수
void calculateSharedMemory(); // shared memory 에 존재하는 값 계산하는 함수
void clearSharedMemory(); // shared memory 비우는 함수

int main()
```

```c
{

    print_menu();

    char input[2];
    int menu;

    fgets(input, sizeof(input), stdin);
    menu = atoi(input);

    struct timespec start, end;
    struct tm realstart, realend;

    switch (menu)
    {
        case 0 : // exit
            exit(0);
            break;

        case 1 : // CFS_DEFAULT
            {
                cpu_set_t set;
                CPU_ZERO(&set);    // CPU 코어 집합 초기화
                CPU_SET(0, &set);  // CPU 코어 0 을 추가 (다른 코어를 추가하려면 여러 번 호출)

                if (sched_setaffinity(getpid(), sizeof(cpu_set_t), &set) == -1) { // CPU 코어 개수 제한
                    perror("Error setting CPU affinity");
                    return 1;
                }

                for(int i = 0 ; i < 21 ; i++){

                    if((pid[i] = fork()) < 0) { // 자식 프로세스 생성
                        printf("fork error\n");
                        exit(1);
                    }
                    else if(pid[i] == 0) {

                        clock_gettime(CLOCK_REALTIME, &start);

                        calculate();

                        clock_gettime(CLOCK_REALTIME, &end);

                        localtime_r((time_t *)&start, &realstart);
                        localtime_r((time_t *)&end, &realend);
```

```
                        printf("PID: %d | Start time : %02d:%02d:%02d.%06ld | End
time : %02d:%02d:%02d.%06ld | Elapsed time : %s\n",
                                getpid(),
                                realstart.tm_hour,realstart.tm_min, realstart.tm_sec,
start.tv_nsec / 1000,
                                realend.tm_hour,realend.tm_min, realend.tm_sec,
end.tv_nsec / 1000,
                                getElapsedTimeSpec(start, end));

                            addElapsedTime(printElapsedTime);
                            exit(0);
                        }
                    }

                    for(int i = 0 ; i < 21 ; i++) {
                        pid_t wpid = waitpid(pid[i],&status,0); // 자식 프로세스 종료
상태를 회수
                    }

                    calculateSharedMemory();

                    printf("Scheduling Policy: CFS_DEFAULT | Average elapsed time:
%06f\n", averageElapsedTime / 21);

                    clearSharedMemory();

                    break;
                }

            case 2 : // CFS_NICE
                {
                    cpu_set_t set;
                    CPU_ZERO(&set);     // CPU 코어 집합 초기화
                    CPU_SET(0, &set);  // CPU 코어 0 을 추가 (다른 코어를 추가하려면 여러 번
호출)


                    if (sched_setaffinity(getpid(), sizeof(cpu_set_t), &set) == -1) {
// CPU 코어 개수 제한
                        perror("Error setting CPU affinity");
                        return 1;
                    }

                    for(int i = 0 ; i < 21 ; i++){
                        if((pid[i] = fork()) < 0) { // 자식 프로세스 생성
                            printf("fork error\n");
                            exit(1);
                        }
                        else if(pid[i] == 0) {

                            if(i < 7) // process 마다 nice 값 부여
```

```
                            nice(19);
                        else if(i < 14)
                            nice(0);
                        else
                            nice(-20);

                        clock_gettime(CLOCK_REALTIME, &start);

                        calculate();

                        clock_gettime(CLOCK_REALTIME, &end);

                        localtime_r((time_t *)&start, &realstart);
                        localtime_r((time_t *)&end, &realend);

                        printf("PID: %d | Start time : %02d:%02d:%02d.%06ld | End
time : %02d:%02d:%02d.%06ld | Elapsed time : %s\n",
                            getpid(),
                            realstart.tm_hour,realstart.tm_min, realstart.tm_sec,
start.tv_nsec / 1000,
                            realend.tm_hour,realend.tm_min, realend.tm_sec,
end.tv_nsec / 1000,
                            getElapsedTimeSpec(start, end));

                        addElapsedTime(printElapsedTime);
                        exit(0);
                    }
                }

                for(int i = 0 ; i < 21 ; i++) {
                    pid_t wpid = waitpid(pid[i],&status,0); // 자식 프로세스 종료
상태를 회수
                }

                calculateSharedMemory();

                printf("Scheduling Policy: CFS_NICE | Average elapsed time:
%06f\n", averageElapsedTime / 21);

                clearSharedMemory();

                break;
            }

        case 3 : // RT_FIFO
            {
                cpu_set_t set;
                CPU_ZERO(&set);     // CPU 코어 집합 초기화
                CPU_SET(0, &set);  // CPU 코어 0 을 추가 (다른 코어를 추가하려면 여러 번
호출)
```

```
                if (sched_setaffinity(getpid(), sizeof(cpu_set_t), &set) == -1) {
// CPU 코어 개수 제한
                    perror("Error setting CPU affinity");
                    return 1;
                }

                for(int i = 0 ; i < 21 ; i++){

                    if((pid[i] = fork()) < 0) { // 자식 프로세스 생성
                        printf("fork error\n");
                        exit(1);
                    }
                    else if(pid[i] == 0) {

                        // FIFO 스케줄러 정책 설정
                        struct sched_param schedParam;
                        schedParam.sched_priority = 50;
                        int policy = SCHED_FIFO;

                        if (sched_setscheduler(getpid(), policy, &schedParam) == -
1) {
                            perror("Error setting scheduler policy");
                            return 1;
                        }

                        clock_gettime(CLOCK_REALTIME, &start);

                        calculate();

                        clock_gettime(CLOCK_REALTIME, &end);

                        localtime_r((time_t *)&start, &realstart);
                        localtime_r((time_t *)&end, &realend);

                        printf("PID: %d | Start time : %02d:%02d:%02d.%06ld | End
time : %02d:%02d:%02d.%06ld | Elapsed time : %s\n",
                            getpid(),
                            realstart.tm_hour,realstart.tm_min, realstart.tm_sec,
start.tv_nsec / 1000,
                            realend.tm_hour,realend.tm_min, realend.tm_sec,
end.tv_nsec / 1000,
                            getElapsedTimeSpec(start, end));

                        addElapsedTime(printElapsedTime);
                        exit(0);
                    }
                }

                for(int i = 0 ; i < 21 ; i++) {
```

```
                    pid_t wpid = waitpid(pid[i],&status,0); // 자식 프로세스 종료
상태를 회수
                }

                calculateSharedMemory();

                printf("Scheduling Policy: RT_FIFO | Average elapsed time: %06f\n",
averageElapsedTime / 21);

                clearSharedMemory();

                break;
            }

        case 4 : // RT_RR
            {

                printf("Input time slice :");

                int num; // time slice 저장하는 변수
                scanf("%d", &num);

                cpu_set_t set;
                CPU_ZERO(&set);    // CPU 코어 집합 초기화
                CPU_SET(0, &set);  // CPU 코어 0 을 추가 (다른 코어를 추가하려면 여러 번
호출)


                if (sched_setaffinity(getpid(), sizeof(cpu_set_t), &set) == -1) {
// CPU 코어 개수 제한
                    perror("Error setting CPU affinity");
                    return 1;
                }

                FILE *fp = fopen("/proc/sys/kernel/sched_rr_timeslice_ms", "w");
                if (fp == NULL) {
                perror("Error opening file");
                return 1;
                }

                // 파일에 timeslice 값을 쓰고 파일을 닫기
                fprintf(fp, "%d", num);
                fclose(fp);

                for(int i = 0 ; i < 21 ; i++){
                    if((pid[i] = fork()) < 0) { // 자식 프로세스 생성
                        printf("fork error\n");
                        exit(1);
                    }
                    else if(pid[i] == 0) {
```

```
                        // Round-Robin 스케줄러 정책 설정
                        struct sched_param schedParam;
                        schedParam.sched_priority = 50;
                        int policy = SCHED_RR;

                        if (sched_setscheduler(getpid(), policy, &schedParam) == -
1) {
                            perror("Error setting scheduler policy");
                            return 1;
                        }

                        clock_gettime(CLOCK_REALTIME, &start);

                        calculate();

                        clock_gettime(CLOCK_REALTIME, &end);

                        localtime_r((time_t *)&start, &realstart);
                        localtime_r((time_t *)&end, &realend);

                        printf("PID: %d | Start time : %02d:%02d:%02d.%06ld | End
time : %02d:%02d:%02d.%06ld | Elapsed time : %s\n",
                            getpid(),
                            realstart.tm_hour,realstart.tm_min, realstart.tm_sec,
start.tv_nsec / 1000,
                            realend.tm_hour,realend.tm_min, realend.tm_sec,
end.tv_nsec / 1000,
                            getElapsedTimeSpec(start, end));

                        addElapsedTime(printElapsedTime);
                        exit(0);
                    }
                }

                for(int i = 0 ; i < 21 ; i++) {
                    pid_t wpid = waitpid(pid[i],&status,0); // 자식 프로세스 종료
상태를 회수
                }

                calculateSharedMemory();

                printf("Scheduling Policy: RT_RR |  Time Quantum: %d ms | Average
elapsed time: %06f\n", num, averageElapsedTime / 21);

                clearSharedMemory();

                break;
            }

        default:
            printf("Wrong input\n");
```

```
            break;
    }



}

// menu 출력하는 함수
void print_menu() {
    printf("Input the Scheduling Policy to apply :\n");
    printf("1. CFS_DEFAULT\n");
    printf("2. CFS_NICE\n");
    printf("3. RT_FIFO\n");
    printf("4. RT_RR\n");
    printf("0. exit\n");
    printf("Input : ");
}

// process 가 실행할 계산하는 함수
void calculate() {
    int count = 0, k, i, j;
    int result[100][100], A[100][100], B[100][100];

    memset(result, 0, sizeof(result));
    memset(A, 0, sizeof(A));
    memset(B, 0, sizeof(B));

    while(count < 100){
        for(k = 0; k < 100; k++){
            for(i = 0; i < 100; i++) {
                for(j = 0; j < 100; j++) {
                    result[k][j] += A[k][i] * B[i][j];
                }
            }
        }
        count++;
    }
}

// elpased time 구하는 함수
char* getElapsedTimeSpec(struct timespec Tstart, struct timespec Tend) {
    Tend.tv_nsec = Tend.tv_nsec - Tstart.tv_nsec;
    Tend.tv_sec = Tend.tv_sec - Tstart.tv_sec;

    Tend.tv_nsec += (Tend.tv_sec*1000000000);

    sprintf(printElapsedTime, "%lf",Tend.tv_nsec / 1000000000.0);
    return printElapsedTime;
}

// elapsed time shared memory 에 저장하는 함수
```

```c
void addElapsedTime(const char *printElapsedTime) {
    // shared memory 세그먼트를 열거나 생성
    shm_fd = shm_open(shm_name, O_RDWR | O_CREAT, 0644);
    if (shm_fd == -1) {
        perror("shm_open");
        return;
    }

    // 새 데이터를 수용할 수 있도록 shared memory 세그먼트의 크기를 조정
    struct stat st;
    if (fstat(shm_fd, &st) == -1) {
        perror("fstat");
        close(shm_fd);
        return;
    }
    off_t new_size = st.st_size + strlen(printElapsedTime) + 1;
    if (ftruncate(shm_fd, new_size) == -1) {
        perror("ftruncate");
        close(shm_fd);
        return;
    }

    // shared memory 세그먼트를 프로세스의 메모리 공간에 매핑
    shm_ptr = mmap(NULL, new_size, PROT_READ | PROT_WRITE, MAP_SHARED, shm_fd, 0);
    if (shm_ptr == MAP_FAILED) {
        perror("mmap");
        close(shm_fd);
        return;
    }

    // `printElapsedTime`을 shared memory 에 추가
    strcpy(shm_ptr + st.st_size, printElapsedTime);
    shm_ptr[new_size - 1] = '\n'; // 끝에 개행 추가

    // 매핑 해제 및 shared memory 세그먼트 닫기
    munmap(shm_ptr, new_size);
    close(shm_fd);
}

// shared memory 에 존재하는 값 계산하는 함수
void calculateSharedMemory() {
    int shm_fd;
    char *shm_ptr, *line;
    const char *shm_name = "monitor_memory";

    // shared memory 세그먼트 열기
    shm_fd = shm_open(shm_name, O_RDONLY, 0);
    if (shm_fd == -1) {
        perror("shm_open");
        return;
```

```
    }

    // shared memory 세그먼트의 크기 얻기
    struct stat st;
    if (fstat(shm_fd, &st) == -1) {
        perror("fstat");
        close(shm_fd);
        return;
    }

    // shared memory 세그먼트를 프로세스의 메모리 공간에 매핑
    shm_ptr = mmap(NULL, st.st_size, PROT_READ, MAP_SHARED, shm_fd, 0);
    if (shm_ptr == MAP_FAILED) {
        perror("mmap");
        close(shm_fd);
        return;
    }

    // 임시 버퍼에 메모리의 내용 복사 (strtok 가 원본을 수정하기 때문에)
    char tempBuffer[st.st_size + 1];
    strncpy(tempBuffer, shm_ptr, st.st_size);
    tempBuffer[st.st_size] = '\0'; // 문자열의 끝을 표시

    // shared memory 세그먼트의 내용을 개행별로 잘라서 출력
    line = strtok(tempBuffer, "\n");
    while(line != NULL) {
        averageElapsedTime += strtod(line, NULL);
        line = strtok(NULL, "\n");
    }

    // 매핑 해제 및 shared memory 세그먼트 닫기
    munmap(shm_ptr, st.st_size);
    close(shm_fd);
}

// shared memory 비우는 함수
void clearSharedMemory() {
    // shared memory 세그먼트 삭제
    if (shm_unlink(shm_name) == -1) {
        perror("shm_unlink");
        return;
    }
}
```

5. 실행 화면 캡쳐

CFS_DEFAULT 스케줄러

```
root@os20192393:/home/os20192393# ./a.out
Input the Scheduling Policy to apply :
1. CFS_DEFAULT
2. CFS_NICE
3. RT_FIFO
4. RT_RR
0. exit
Input : 1
PID: 2369 | Start time : 12:59:55.440197 | End time : 13:00:00.207546 | Elapsed time : 4.767349
PID: 2382 | Start time : 12:59:55.420241 | End time : 13:00:00.218505 | Elapsed time : 4.798264
PID: 2368 | Start time : 12:59:55.444183 | End time : 13:00:00.222880 | Elapsed time : 4.778697
PID: 2376 | Start time : 12:59:55.396278 | End time : 13:00:00.225991 | Elapsed time : 4.829712
PID: 2374 | Start time : 12:59:55.388269 | End time : 13:00:00.229682 | Elapsed time : 4.841413
PID: 2373 | Start time : 12:59:55.384347 | End time : 13:00:00.233490 | Elapsed time : 4.849144
PID: 2378 | Start time : 12:59:55.404254 | End time : 13:00:00.236877 | Elapsed time : 4.832623
PID: 2381 | Start time : 12:59:55.416256 | End time : 13:00:00.241487 | Elapsed time : 4.825230
PID: 2367 | Start time : 12:59:55.448180 | End time : 13:00:00.246421 | Elapsed time : 4.798240
PID: 2385 | Start time : 12:59:55.432195 | End time : 13:00:00.247827 | Elapsed time : 4.815632
PID: 2377 | Start time : 12:59:55.400267 | End time : 13:00:00.250190 | Elapsed time : 4.849923
PID: 2383 | Start time : 12:59:55.424224 | End time : 13:00:00.252381 | Elapsed time : 4.828156
PID: 2380 | Start time : 12:59:55.412256 | End time : 13:00:00.255694 | Elapsed time : 4.843438
PID: 2384 | Start time : 12:59:55.428211 | End time : 13:00:00.257164 | Elapsed time : 4.828953
PID: 2375 | Start time : 12:59:55.392296 | End time : 13:00:00.259678 | Elapsed time : 4.867383
PID: 2379 | Start time : 12:59:55.408255 | End time : 13:00:00.260787 | Elapsed time : 4.852532
PID: 2372 | Start time : 12:59:55.380330 | End time : 13:00:00.262992 | Elapsed time : 4.882661
PID: 2370 | Start time : 12:59:55.436192 | End time : 13:00:00.199830 | Elapsed time : 4.763639
PID: 2371 | Start time : 12:59:55.375667 | End time : 13:00:00.264535 | Elapsed time : 4.888867
PID: 2366 | Start time : 12:59:55.520167 | End time : 13:00:00.268004 | Elapsed time : 4.747837
PID: 2365 | Start time : 12:59:55.596184 | End time : 13:00:00.274463 | Elapsed time : 4.678280
Scheduling Policy: CFS_DEFAULT | Average elapsed time: 4.817523
```

CFS_NICE 스케줄러

```
root@os20192393:/home/os20192393# ./a.out
Input the Scheduling Policy to apply :
1. CFS_DEFAULT
2. CFS_NICE
3. RT_FIFO
4. RT_RR
0. exit
Input : 2
PID: 2571 | NICE : -20 | Start time : 13:14:48.112172 | End time : 13:14:49.606730 | Elapsed time : 1.494558
PID: 2570 | NICE : -20 | Start time : 13:14:48.096204 | End time : 13:14:49.618168 | Elapsed time : 1.521964
PID: 2572 | NICE : -20 | Start time : 13:14:48.128164 | End time : 13:14:49.646623 | Elapsed time : 1.518459
PID: 2569 | NICE : -20 | Start time : 13:14:48.085147 | End time : 13:14:49.651306 | Elapsed time : 1.566159
PID: 2573 | NICE : -20 | Start time : 13:14:48.140183 | End time : 13:14:49.654353 | Elapsed time : 1.514169
PID: 2568 | NICE : -20 | Start time : 13:14:48.064269 | End time : 13:14:49.660019 | Elapsed time : 1.595750
PID: 2567 | NICE : -20 | Start time : 13:14:48.028599 | End time : 13:14:49.680531 | Elapsed time : 1.651932
PID: 2560 | NICE : 0 | Start time : 13:14:48.000326 | End time : 13:14:51.274687 | Elapsed time : 3.274361
PID: 2561 | NICE : 0 | Start time : 13:14:48.004262 | End time : 13:14:51.279434 | Elapsed time : 3.275172
PID: 2565 | NICE : 0 | Start time : 13:14:48.020286 | End time : 13:14:51.284511 | Elapsed time : 3.264225
PID: 2563 | NICE : 0 | Start time : 13:14:48.012280 | End time : 13:14:51.291237 | Elapsed time : 3.278957
PID: 2566 | NICE : 0 | Start time : 13:14:48.024275 | End time : 13:14:51.297283 | Elapsed time : 3.273007
PID: 2562 | NICE : 0 | Start time : 13:14:48.008247 | End time : 13:14:51.302916 | Elapsed time : 3.294668
PID: 2564 | NICE : 0 | Start time : 13:14:48.016518 | End time : 13:14:51.304062 | Elapsed time : 3.287545
PID: 2556 | NICE : 19 | Start time : 13:14:49.124190 | End time : 13:14:52.883654 | Elapsed time : 3.759464
PID: 2553 | NICE : 19 | Start time : 13:14:49.744165 | End time : 13:14:52.875875 | Elapsed time : 3.131711
PID: 2558 | NICE : 19 | Start time : 13:14:48.284177 | End time : 13:14:52.900413 | Elapsed time : 4.616237
PID: 2554 | NICE : 19 | Start time : 13:14:49.740197 | End time : 13:14:52.900782 | Elapsed time : 3.160585
PID: 2555 | NICE : 19 | Start time : 13:14:49.680935 | End time : 13:14:52.904670 | Elapsed time : 3.223735
PID: 2557 | NICE : 19 | Start time : 13:14:48.624186 | End time : 13:14:52.910260 | Elapsed time : 4.286074
PID: 2559 | NICE : 19 | Start time : 13:14:47.994203 | End time : 13:14:52.916035 | Elapsed time : 4.921833
Scheduling Policy: CFS_NICE | Average elapsed time: 2.900503
```

RT_FIFO 스케줄러

```
root@os20192393:/home/os20192393# ./a.out
Input the Scheduling Policy to apply :
1. CFS_DEFAULT
2. CFS_NICE
3. RT_FIFO
4. RT_RR
0. exit
Input : 3
PID: 2581 | Start time : 13:14:56.852217 | End time : 13:14:57.121227 | Elapsed time : 0.269011
PID: 2582 | Start time : 13:14:57.121669 | End time : 13:14:57.354494 | Elapsed time : 0.232826
PID: 2583 | Start time : 13:14:57.354953 | End time : 13:14:57.588289 | Elapsed time : 0.233336
PID: 2585 | Start time : 13:14:57.588742 | End time : 13:14:57.821289 | Elapsed time : 0.232548
PID: 2586 | Start time : 13:14:57.821755 | End time : 13:14:58.056748 | Elapsed time : 0.234993
PID: 2587 | Start time : 13:14:58.057138 | End time : 13:14:58.336045 | Elapsed time : 0.278907
PID: 2588 | Start time : 13:14:58.128205 | End time : 13:14:58.566026 | Elapsed time : 0.437821
PID: 2589 | Start time : 13:14:58.132303 | End time : 13:14:58.797247 | Elapsed time : 0.664943
PID: 2584 | Start time : 13:14:58.136196 | End time : 13:14:59.027841 | Elapsed time : 0.891645
PID: 2590 | Start time : 13:14:58.140220 | End time : 13:14:59.433323 | Elapsed time : 1.293103
PID: 2591 | Start time : 13:14:58.144173 | End time : 13:14:59.660496 | Elapsed time : 1.516324
PID: 2592 | Start time : 13:14:58.148179 | End time : 13:14:59.887626 | Elapsed time : 1.739447
PID: 2593 | Start time : 13:14:58.152164 | End time : 13:15:00.115529 | Elapsed time : 1.963365
PID: 2594 | Start time : 13:14:58.156164 | End time : 13:15:00.414239 | Elapsed time : 2.258074
PID: 2580 | Start time : 13:14:58.160166 | End time : 13:15:00.642943 | Elapsed time : 2.482777
PID: 2595 | Start time : 13:14:58.164164 | End time : 13:15:00.874662 | Elapsed time : 2.710498
PID: 2579 | Start time : 13:14:58.168163 | End time : 13:15:01.106529 | Elapsed time : 2.938366
PID: 2578 | Start time : 13:14:58.172173 | End time : 13:15:01.392007 | Elapsed time : 3.219834
PID: 2577 | Start time : 13:14:58.176163 | End time : 13:15:01.620309 | Elapsed time : 3.444146
PID: 2576 | Start time : 13:14:59.080201 | End time : 13:15:01.845647 | Elapsed time : 2.765446
PID: 2575 | Start time : 13:14:59.084171 | End time : 13:15:02.071220 | Elapsed time : 2.987049
Scheduling Policy: RT_FIFO | Average elapsed time: 1.561641
```

RT_RR 스케줄러 (time slice = 10 ms)

```
root@os20192393:/home/os20192393# ./a.out
Input the Scheduling Policy to apply :
1. CFS_DEFAULT
2. CFS_NICE
3. RT_FIFO
4. RT_RR
0. exit
Input : 4
Input time slice :10
PID: 2606 | Start time : 13:16:09.210256 | End time : 13:16:09.477249 | Elapsed time : 0.266993
PID: 2607 | Start time : 13:16:09.477743 | End time : 13:16:09.711324 | Elapsed time : 0.233581
PID: 2608 | Start time : 13:16:09.711765 | End time : 13:16:09.954330 | Elapsed time : 0.242565
PID: 2609 | Start time : 13:16:09.954764 | End time : 13:16:10.285596 | Elapsed time : 0.330831
PID: 2613 | Start time : 13:16:10.176176 | End time : 13:16:11.099416 | Elapsed time : 0.923240
PID: 2610 | Start time : 13:16:10.164220 | End time : 13:16:11.104985 | Elapsed time : 0.940765
PID: 2611 | Start time : 13:16:10.168185 | End time : 13:16:11.107331 | Elapsed time : 0.939145
PID: 2612 | Start time : 13:16:10.172182 | End time : 13:16:11.108479 | Elapsed time : 0.936297
PID: 2614 | Start time : 13:16:11.108776 | End time : 13:16:14.289029 | Elapsed time : 3.180254
PID: 2615 | Start time : 13:16:11.112174 | End time : 13:16:14.295019 | Elapsed time : 3.182845
PID: 2616 | Start time : 13:16:11.116177 | End time : 13:16:14.303139 | Elapsed time : 3.186962
PID: 2618 | Start time : 13:16:11.124183 | End time : 13:16:14.323609 | Elapsed time : 3.199427
PID: 2620 | Start time : 13:16:11.132234 | End time : 13:16:14.339208 | Elapsed time : 3.206973
PID: 2605 | Start time : 13:16:11.136180 | End time : 13:16:14.344122 | Elapsed time : 3.207942
PID: 2604 | Start time : 13:16:11.140294 | End time : 13:16:14.347186 | Elapsed time : 3.206892
PID: 2603 | Start time : 13:16:11.144230 | End time : 13:16:14.348847 | Elapsed time : 3.204617
PID: 2602 | Start time : 13:16:11.148183 | End time : 13:16:14.350141 | Elapsed time : 3.201958
PID: 2601 | Start time : 13:16:11.152171 | End time : 13:16:14.352452 | Elapsed time : 3.200281
PID: 2600 | Start time : 13:16:11.156192 | End time : 13:16:14.353785 | Elapsed time : 3.197592
PID: 2617 | Start time : 13:16:11.120174 | End time : 13:16:14.358481 | Elapsed time : 3.238307
PID: 2619 | Start time : 13:16:11.128179 | End time : 13:16:14.360643 | Elapsed time : 3.232464
Scheduling Policy: RT_RR |  Time Quantum: 10 ms | Average elapsed time: 2.212378
```

RT_RR 스케줄러 (time slice = 100 ms)

```
root@os20192393:/home/os20192393# ./a.out
Input the Scheduling Policy to apply :
1. CFS_DEFAULT
2. CFS_NICE
3. RT_FIFO
4. RT_RR
0. exit
Input : 4
Input time slice :100
PID: 2628 | Start time : 13:16:50.123955 | End time : 13:16:50.390195 | Elapsed time : 0.266240
PID: 2629 | Start time : 13:16:50.390696 | End time : 13:16:50.622830 | Elapsed time : 0.232134
PID: 2630 | Start time : 13:16:50.623280 | End time : 13:16:50.855599 | Elapsed time : 0.232319
PID: 2632 | Start time : 13:16:50.856041 | End time : 13:16:51.089466 | Elapsed time : 0.233425
PID: 2631 | Start time : 13:16:51.089880 | End time : 13:16:54.517805 | Elapsed time : 3.427925
PID: 2633 | Start time : 13:16:51.128220 | End time : 13:16:54.547996 | Elapsed time : 3.419776
PID: 2634 | Start time : 13:16:51.132193 | End time : 13:16:54.580715 | Elapsed time : 3.448523
PID: 2635 | Start time : 13:16:51.136184 | End time : 13:16:54.618131 | Elapsed time : 3.481947
PID: 2636 | Start time : 13:16:51.140187 | End time : 13:16:54.663978 | Elapsed time : 3.523791
PID: 2637 | Start time : 13:16:51.144174 | End time : 13:16:54.697087 | Elapsed time : 3.552913
PID: 2627 | Start time : 13:16:51.148182 | End time : 13:16:54.731139 | Elapsed time : 3.582957
PID: 2638 | Start time : 13:16:51.152168 | End time : 13:16:54.767074 | Elapsed time : 3.614906
PID: 2639 | Start time : 13:16:51.156184 | End time : 13:16:54.799265 | Elapsed time : 3.643081
PID: 2640 | Start time : 13:16:51.160177 | End time : 13:16:55.037018 | Elapsed time : 3.876841
PID: 2641 | Start time : 13:16:51.164251 | End time : 13:16:55.077458 | Elapsed time : 3.913207
PID: 2626 | Start time : 13:16:51.168304 | End time : 13:16:55.112700 | Elapsed time : 3.944396
PID: 2642 | Start time : 13:16:51.172263 | End time : 13:16:55.203608 | Elapsed time : 4.031344
PID: 2625 | Start time : 13:16:51.176175 | End time : 13:16:55.233816 | Elapsed time : 4.057641
PID: 2624 | Start time : 13:16:51.180175 | End time : 13:16:55.261267 | Elapsed time : 4.081092
PID: 2623 | Start time : 13:16:52.080235 | End time : 13:16:55.309045 | Elapsed time : 3.228810
PID: 2622 | Start time : 13:16:52.084215 | End time : 13:16:55.338978 | Elapsed time : 3.254764
Scheduling Policy: RT_RR |  Time Quantum: 100 ms | Average elapsed time: 3.002287
```

RT_RR 스케줄러 (time slice = 1000 ms)

```
root@os20192393:/home/os20192393# ./a.out
Input the Scheduling Policy to apply :
1. CFS_DEFAULT
2. CFS_NICE
3. RT_FIFO
4. RT_RR
0. exit
Input : 4
Input time slice :1000
PID: 2653 | Start time : 13:17:44.960053 | End time : 13:17:45.229173 | Elapsed time : 0.269121
PID: 2654 | Start time : 13:17:45.229638 | End time : 13:17:45.463495 | Elapsed time : 0.233857
PID: 2655 | Start time : 13:17:45.463947 | End time : 13:17:45.696654 | Elapsed time : 0.232707
PID: 2657 | Start time : 13:17:45.697100 | End time : 13:17:45.929376 | Elapsed time : 0.232276
PID: 2659 | Start time : 13:17:45.929836 | End time : 13:17:46.211511 | Elapsed time : 0.281676
PID: 2660 | Start time : 13:17:46.132231 | End time : 13:17:46.451243 | Elapsed time : 0.319013
PID: 2661 | Start time : 13:17:46.136202 | End time : 13:17:46.681556 | Elapsed time : 0.545354
PID: 2662 | Start time : 13:17:46.140198 | End time : 13:17:46.910735 | Elapsed time : 0.770537
PID: 2663 | Start time : 13:17:46.144182 | End time : 13:17:47.253882 | Elapsed time : 1.109699
PID: 2664 | Start time : 13:17:46.148192 | End time : 13:17:47.483389 | Elapsed time : 1.335197
PID: 2665 | Start time : 13:17:46.152196 | End time : 13:17:47.714190 | Elapsed time : 1.561994
PID: 2666 | Start time : 13:17:46.156193 | End time : 13:17:47.943574 | Elapsed time : 1.787381
PID: 2667 | Start time : 13:17:46.160201 | End time : 13:17:48.247818 | Elapsed time : 2.087617
PID: 2656 | Start time : 13:17:46.164187 | End time : 13:17:48.477005 | Elapsed time : 2.312818
PID: 2658 | Start time : 13:17:46.168172 | End time : 13:17:48.706701 | Elapsed time : 2.538529
PID: 2652 | Start time : 13:17:46.172179 | End time : 13:17:48.937183 | Elapsed time : 2.765004
PID: 2651 | Start time : 13:17:46.176180 | End time : 13:17:49.228080 | Elapsed time : 3.051900
PID: 2650 | Start time : 13:17:47.084222 | End time : 13:17:49.458336 | Elapsed time : 2.374113
PID: 2649 | Start time : 13:17:47.088173 | End time : 13:17:49.687113 | Elapsed time : 2.598940
PID: 2648 | Start time : 13:17:47.092179 | End time : 13:17:49.916380 | Elapsed time : 2.824200
PID: 2647 | Start time : 13:17:47.096176 | End time : 13:17:50.200562 | Elapsed time : 3.104387
Scheduling Policy: RT_RR |  Time Quantum: 1000 ms | Average elapsed time: 1.539825
```

시스템 종료

```
root@os20192393:/home/os20192393# ./a.out
Input the Scheduling Policy to apply :
1. CFS_DEFAULT
2. CFS_NICE
3. RT_FIFO
4. RT_RR
0. exit
Input : 0
root@os20192393:/home/os20192393#
```

6. 실행 결과 분석 내용

CFS_DEFAULT 정책을 적용했을 경우 completely fair scheduler 로 공평성 유지해서 linux 기본 스케줄링 따라서 실행되는 것을 확인할 수 있다.

CFS_NICE 정책을 적용했을 경우 Nice 값이 -20 인 process 7 개 먼저 실행 후 출력 이후 Nice 값이 0 인 process 7 개 실행 후 출력 이후 Nice 값이 19 인 process 7 개 실행 후 출력 된다. 먼저 생성된 process 일지라도 nice 값에 따라 nice 값이 낮은 process 들이 높은 우선순위를 가져 먼저 실행되는 것을 확인할 수 있다.

RT_FIFO 정책을 적용했을 경우 process 생성 후 실행하는 순서대로 끝나는 것을 확인할 수 있다.

RT_RR 정책을 적용했을 경우 time slice 의 크기에 따라 평균 elapsed time 의 차이를 볼 수 있다. Time slice 값을 적게 적용한 경우 process 마다 context switching 을 많이 하게 되어 overhead 가 많이 발생해서 평균 elapsed time 이 더 오래 걸리는 것을 확인할 수 있다.

7. 가산점 작업 설명

리눅스 커널 코드의 수정을 통해 RT_FIFO 스케줄링 정책으로 변경하면서 생성되는 프로세스들의 CPU 점유 시간 누적 값을 프로세스 종료 시 커널 로그에 출력하도록 했다.

core.c 파일 내에서 sched_fork 함수 내부 코드 수정한 부분

```
// SCHED_FIFO
if(p->policy == SCHED_NORMAL){ // 현재 프로세스 정책 확인
    p->prio = current->normal_prio - NICE_WIDTH -
        PRIO_TO_NICE(current->static_prio); // 프로세스 동적 우선순위 바꿔주기
    p->normal_prio = p->prio; // 프로세스 일반 순위를 동적 우선순위로 바꿔주기
    p->rt_priority = p->prio; // 프로세스 실시간 순위를 동적 우선순위로 바꿔주기
    p->policy = SCHED_FIFO; // 프로세스 정책 SCHED_FIFO 로 변경
    p->static_prio = NICE_TO_PRIO(0);
}
```

새로운 프로세스 생성시 sched_fork 함수 호출되는데 이때 프로세스 스케줄링 정책을 변경하도록 했다.

SCHED_NORMAL 인 경우 SCHED_FIFO 로 정책 변경하는 코드

core.c 파일 내에서 _sched_setscheduler 함수 내부 코드 수정한 부분

```
// SCHED_FIFO
if (attr.sched_policy == SCHED_NORMAL) { // 현재 프로세스 정책 확인
    attr.sched_priority = param->sched_priority -
        NICE_WIDTH - attr.sched_nice; // 프로세스 우선순위 바꿔주기
    attr.sched_policy = SCHED_FIFO;  // 프로세스 정책 바꿔주기
}
```

SCHED_NORMAL 인 경우 SCHED_FIFO 로 정책 변경하는 코드

exit.c 파일 내에서 do_exit 함수 내부 코드 수정한 부분

```
    // 프로세스의 CPU 시간을 가져옵니다.
    unsigned long long total_time = tsk->utime + tsk->stime;

    // 로그 메시지를 출력합니다.
    printk("Process %d (%s) terminated with total CPU time: %llu\n", tsk->pid, tsk-
>comm, total_time);
```

프로세스 종료할 때 호출되는 함수로 함수에서 사용하는 task_strurct 구조체 내에 있는 utime, stime 을 이용하여 cpu burst time 출력했다.

로그 메시지가 정확한지 확인하기 위해 pid 값과 해당 프로세스 실행 명령어도 같이 출력해서 확인했다.

test.c 커널 코드 수정된거 확인하는 코드

```
#define _GNU_SOURCE
#include <sched.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <signal.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <sys/time.h>
#include <sys/mman.h>
#include <sys/resource.h>
#include <string.h>
#include <time.h>
#include <errno.h>
#include <signal.h>
#include <dirent.h>
#include <sys/wait.h>
#include <sys/syscall.h>
#include <errno.h>
#include <stdint.h>

int status;
int pid[21];

void calculate(); // process 가 실행할 계산하는 함수

int main()
{
    cpu_set_t set;
```

```
    CPU_ZERO(&set);    // CPU 코어 집합 초기화
    CPU_SET(0, &set);  // CPU 코어 0을 추가 (다른 코어를 추가하려면 여러 번 호출)

    if (sched_setaffinity(getpid(), sizeof(cpu_set_t), &set) == -1) { // CPU 코어
개수 제한
        perror("Error setting CPU affinity");
        return 1;
    }

    for(int i = 0 ; i < 21 ; i++){

        if((pid[i] = fork()) < 0) { // 자식 프로세스 생성
            printf("fork error\n");
            exit(1);
        }
        else if(pid[i] == 0) {

            calculate();

            exit(0);
        }
    }

    for(int i = 0 ; i < 21 ; i++) {
        pid_t wpid = waitpid(pid[i],&status,0); // 자식 프로세스 종료 상태를 회수
    }

}

// process 가 실행할 계산하는 함수
void calculate() {
    int count = 0, k, i, j;
    int result[100][100], A[100][100], B[100][100];

    memset(result, 0, sizeof(result));
    memset(A, 0, sizeof(A));
    memset(B, 0, sizeof(B));

    while(count < 100){
        for(k = 0; k < 100; k++){
            for(i = 0; i < 100; i++) {
                for(j = 0; j < 100; j++) {
                    result[k][j] += A[k][i] * B[i][j];
                }
            }
        }
        count++;
    }
}
```

8. 가산점 코드 실행 화면 캡쳐

RT_FIFO 스케줄링 정책 변경된 것 확인

CLS 부분이 모두 FF 로 변경된 것 확인

```
root@os20192393:/home/os20192393# ps -c
    PID CLS PRI TTY          TIME CMD
   2121 FF   59 pts/0    00:00:00 sudo
   2122 FF   59 pts/0    00:00:00 bash
   9722 FF   59 pts/0    00:00:00 ps
root@os20192393:/home/os20192393#
```

생성되는 되는 각 프로세스의 CPU 점유 시간 누적 값을 프로세스 종료 시 커널 로그에 출력

```
[12073.714119] Process 5049 (gmain) terminated with total CPU time: 0
[12073.714201] Process 5050 (gdbus) terminated with total CPU time: 4000000
[12073.714272] Process 5051 (dconf worker) terminated with total CPU time: 0
[12073.714284] Process 5063 (pool-tracker-ex) terminated with total CPU time: 8000000
[12076.664381] Process 5076 (a.out) terminated with total CPU time: 268000000
[12076.899698] Process 5077 (a.out) terminated with total CPU time: 236000000
[12077.131903] Process 5078 (a.out) terminated with total CPU time: 232000000
[12077.362007] Process 5079 (a.out) terminated with total CPU time: 228000000
[12077.591245] Process 5080 (a.out) terminated with total CPU time: 228000000
[12077.873744] Process 5081 (a.out) terminated with total CPU time: 240000000
[12078.102597] Process 5082 (a.out) terminated with total CPU time: 228000000
[12078.332004] Process 5083 (a.out) terminated with total CPU time: 232000000
[12078.562538] Process 5084 (a.out) terminated with total CPU time: 228000000
[12078.850760] Process 5085 (a.out) terminated with total CPU time: 240000000
[12079.079661] Process 5086 (a.out) terminated with total CPU time: 232000000
[12079.308618] Process 5087 (a.out) terminated with total CPU time: 228000000
[12079.537630] Process 5088 (a.out) terminated with total CPU time: 228000000
[12079.824104] Process 5089 (a.out) terminated with total CPU time: 240000000
[12080.052925] Process 5090 (a.out) terminated with total CPU time: 228000000
[12080.282062] Process 5091 (a.out) terminated with total CPU time: 228000000
[12080.510658] Process 5092 (a.out) terminated with total CPU time: 228000000
[12080.801201] Process 5093 (a.out) terminated with total CPU time: 240000000
[12081.030941] Process 5094 (a.out) terminated with total CPU time: 228000000
[12081.260006] Process 5095 (a.out) terminated with total CPU time: 232000000
[12081.489242] Process 5096 (a.out) terminated with total CPU time: 228000000
[12081.489862] Process 5075 (a.out) terminated with total CPU time: 4000000
[12090.203663] Process 5072 (pool-tracker-mi) terminated with total CPU time: 0
[12095.401775] Process 5098 (a.out) terminated with total CPU time: 268000000
[12095.632920] Process 5099 (a.out) terminated with total CPU time: 232000000
[12095.706801] Process 5119 (wal-checkpoint) terminated with total CPU time: 0
```