

곽민석 - 20183380
2023년 11월 6일 월요일
가산점 수행 여부: X

운영체제

스케줄러 변경에 대한 자식 프로세스 실행과 결과

과제 정보

1. 파일 목록

파일명	설명
Makefile	응용프로그램의 컴파일을 위한 Makefile.
main.c	스케줄러 변경 및 프로그램 실행을 위한 소스 파일.

2. 주의사항



RT RR 스케줄러를 이용하기 전 다음의 명령어를
이용하여 Time Slice를 정의 후 실행해 주시기 바랍니다.

```
sudo sysctl kernel.sched_rr_timeslice_ms=N
```

명령어 1. Time Slice 변경 명령어, N은 ms 단위의 자연수.

또한 응용프로그램을 실행할 때
root 권한을 이용해실행시켜 주시기 바랍니다.

```
sudo ./sched
```

명령어 2. root 권한을 이용한 응용프로그램 실행.

1. 과제 설명

리눅스 기본 커널에서는 기본적으로 CFS(이하 Completely Fair Scheduler)를 스케줄러로 이용하며, RT FIFO(이하 Real Time First in First Out), RT RR(이하 Real Time Round Robin)도 사용이 가능하다. 이 기능을 이용하여 각 스케줄러를 이용하여 21개의 자식 프로세스를 실행해보고 결과를 비교해보는 과제이다. 각 비교군은 다음과 같다.

첫 번째로 CFS를 이용하여 Nice 값을 변경하지 않고 실행하는 방법이다. 이때 Nice 기본값은 0으로 설정되어 프로세스가 동작하게 된다.

두 번째로 CFS를 이용하여 각 7개마다 Nice 값을 차등 배정하여 실행하는 방법이다. 7개 각각 -20, 0, 19를 Nice 값으로 설정하여 실행시키고 결과를 확인하게 된다.

세 번째로 RT FIFO를 이용하여 실행하는 방법이다. 이때는 따로 변경사항이 들어가지 않는다.

마지막으로 RT RR을 이용하여 실행하는 방법이다. 이때 Time Slice를 각각 10, 100, 1000ms로 변경하여 실행시켜보고 이들의 각각의 차이점을 비교해본다.

2. 설치 및 실행 방법

A. Application Compile and Execution

1. 프로젝트 폴더에서 “make” 명령어를 이용하여 컴파일 한다.
2. “root 권한”을 이용하여 “./sched”를 실행한다.

B. CFS_DEFAULT

1. 기본 CFS를 이용하여 프로세스를 실행하는 방법이다.
2. 옵션 입력시 1을 입력하면 된다.

C. CFS_NICE

1. 기본 CFS를 이용하여 프로세스를 실행하게 되는데, 7개의 프로세스마다 각각 -20, 0, 19로 Nice 값을 할당하여 실행하는 방법이다. 먼저 실행된 프로세스일수록 높은 Nice 값을 가지도록 설정하였다.
2. 옵션 입력시 2을 입력하면 된다.

D. RT_FIFO

1. RT FIFO를 이용하여 프로세스를 실행하는 방법이다.
2. 옵션 입력시 3을 입력하면 된다.

E. RT_RR

1. RT RR을 이용하여 프로세스를 실행하는 방법이다. Time Slice를 변경하는 방법은 아래의 명령어를 이용하여 터미널에 입력한다. N은 ms 단위의 정수이다.

```
sudo sysctl kernel.sched_rr_timeslice_ms=N
```

명령어 1. Time Slice 변경 명령어.

2. 위의 명령어를 이용하여 Time Slice 변경 후, 응용프로그램에서 옵션 입력시 4을 입력하면 된다.

F. EXIT

1. 옵션 입력시 0을 입력하면 된다.

4. 코드 설명

A. printMenu

```
// 실행 옵션을 출력하기 위한 함수.  
void printMenu() {  
    printf("Input the Scheduling Polity 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");  
}
```

해당 함수는 메뉴를 출력하는 함수이다.

B. printTime

```
// 시간을 "HH:MM:SS.XXXXXX" 포맷으로 출력하는 함수.  
void printTime(struct timeval tv) {  
    struct tm tmStruct;  
    time_t unixTime = tv.tv_sec;  
    gmtime_r(&unixTime, &tmStruct);  
  
    // "HH:MM:SS.XXXXXX" 포맷으로 변경.  
    char timeString[20];  
    strftime(timeString, sizeof(timeString), "%H:%M:%S", &tmStruct);  
  
    // ms 단위로 변환,  
    int milliseconds = tv.tv_usec;  
  
    snprintf(timeString + 8, 8, "%.06d", milliseconds);  
  
    printf("%s", timeString);  
}
```

해당 함수는 timeval 구조체를 받아 “HH:MM:SS.ssssss” 형태로 출력하는 함수이다.

C. printInfo

```
// 실행 정보 출력을 위한 함수.  
void printInfo(int option, int pid, int nice, struct timeval time_start,  
               struct timeval time_end) {  
    double execution_time = (time_end.tv_sec - time_start.tv_sec)  
        + (time_end.tv_usec - time_start.tv_usec) / 1000000.0;  
  
    if(option <= 2)  
        printf("PID: %d | NICE: %d | start time: ", pid, nice);  
    else  
        printf("PID: %d | start time: ", pid);  
  
    printTime(time_start);  
    printf(" | End time: ");  
    printTime(time_end);  
    printf(" | Elapsed time: %.6lf\n", execution_time);  
}
```

해당 함수는 실행이 모두 끝난 후 PID, 프로세스 시작, 종료시간, 작업 수행까지 걸린 시간을 출력하는 함수이다. 실행된 작업이 CFS를 이용하여 수행된 경우 해당 작업의 Nice 값이 같이 출력된다.

D. benchFunction

```
// 실행시 걸린 시간 반환 함수.
void benchFunction() {
    int which = PRIO_PROCESS;
    int result[100][100];
    int A[100][100];
    int B[100][100];
    int count = 0;
    int i, j, k;

    time_t time_start, time_end;
    cpu_set_t set;

    // CPU 코어 설정.
    CPU_ZERO(&set);
    CPU_SET(1, &set);
    sched_setaffinity(getpid(), sizeof(cpu_set_t), &set);

    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++;
    }
}
```

해당 함수는 프로세스의 연산 작업을 시키기 위해 제작한 함수이다. 해당 함수에서는 100 X 100 크기의 두 행렬에 대해 행렬곱을 100번 수행하게 된다. 또한 프로세서의 갯수가 실행시간에 영향을 주지 않도록 함수 sched_setaffinity를 이용하여 실행된 프로세스에 대해 affinity를 설정하여 하나의 코어만 이용하도록 하였다.

E. runCFSNice

```
// CFS Nice 구동을 위한 함수.
void runCFSNice(int i) {
    int which = PRIO_PROCESS;
    if(i / 7 == 2) setpriority(which, 0, -20);
    else if(i / 7 == 1) setpriority(which, 0, 0);
    else if(i / 7 == 0) setpriority(which, 0, 19);
    else printf("%d\n", i / 7);
}
```

해당 함수는 CFS를 이용하여 프로세스마다 Nice 값을 차등적으로 적용하는 함수이다. 늦게 실행된 프로세스여도 낮은 Nice 값을 가지게 되면 어떠한 결과를 내게 되는지 알아보기 위해 먼저 실행되는 프로세스일수록 높은 Nice 값을 설정하도록 하였다.

F. runFIFO

```
// FIFO 구동을 위한 함수.
void runFIFO() {
    int policy = SCHED_FIFO;
    struct sched_param param;

    param.sched_priority = 99;

    sched_setscheduler(0, policy, &param);
}
```

해당 함수는 RT FIFO로 스케줄러를 설정하는 함수이다. RT FIFO는 라이브러리 sched.h에 정의된 상수 SCHED_FIFO를 이용하여 함수 sched_setscheduler를 통해 설정하였다. 또한 우선순위와 상관 없이 수행되도록 우선순위를 99로 고정하였다.

G. runRR

```
// RR 구동을 위한 함수.
void runRR() {
    int policy = SCHED_RR;
    struct sched_param param;

    param.sched_priority = 99;

    sched_setscheduler(0, policy, &param);
}
```

해당 함수는 RT RR로 스케줄러를 설정하는 함수이다. RT RR는 라이브러리 sched.h에 정의된 상수 SCHED_RR를 이용하여 함수 sched_setscheduler를 통해 설정하였다. 또한 우선순위와 상관 없이 수행되도록 우선순위를 99로 고정하였다.

H. getTimeSlice

```
// 현재 타임 슬라이스를 반환하는 함수.
int getTimeSlice() {
    FILE* fp = fopen("/proc/sys/kernel/sched_rr_timeslice_ms", "r");
    int timeslice;

    if(fscanf(fp, "%d", &timeslice) != 1) {
        perror("fscanf");
        fclose(fp);
        return 1;
    }

    fclose(fp);

    return timeslice;
}
```

해당 함수는 RT RR로 실행되었을 때 결과에 시스템에서 설정된 Time Slice를 출력하기 위해 정의된 Time Slice를 가져오는 함수이다. RR의 Time Slice 값은 아래의 파일에 정의되어있다.

/proc/sys/kernel/sched_rr_timeslice_ms

경로 1. Time Slice 정의 파일 경로.

해당 파일의 Time Slice는 ms 단위로 저장되어있다.

I. main

```
int main() {
    int    which;
    int    menu;
    int    shmid;
    pid_t  child_pids;
    key_t  key;

    menu      = 1;
    which     = PRIO_PROCESS;
    key       = ftok("result_space", 65);
    shmid     = shmget(key, 21 * sizeof(int), IPC_CREAT | 0666);

    // Shared Memory 생성.
    int      *shared_memory = shmat(shmid, NULL, 0);
    ...
}
```

해당 부분에서는 먼저 필요한 변수들을 정의한다. 특히 응용프로그램은 자식 프로세스에서 실행된 시간을 받아 평균을 내야하기 때문에 공유 메모리를 두어 결과를 받을 수 있도록 공유 메모리를 선언한다.

```

// Main 함수.
int main() {
    ...
    while(menu) {
        // 메뉴 출력 및 선택 받기.
        printMenu();
        scanf("%d", &menu);

        // 옵션에 따른 기본 설정.
        switch(menu) {
            case 0:
                printf("EXIT\n");
                exit(0);
                break;
            case 1:
            case 2:
            case 3:
            case 4:
                break;
            default:
                printf("Wrong option. \n");
        }
        ...
    }
}

```

이후 선택된 메뉴 번호에 따라 동작하도록 한다. “0”이 들어오게 되면 종료, 옵션 이외에 값은 오류 메시지를 출력하도록 한다.


```

int main() {
    while(menu) {
        ...
        if(menu >= 1 && menu <= 4) {
            // 자식 프로세스 생성 후 실행.
            for(int i = 0; i < 21; i++) {
                child_pids = fork();

                if(child_pids == 0) {
                    struct timeval time_start, time_end;
                    // 시작시간 기록.
                    gettimeofday(&time_start, NULL);

                    // 옵션에 맞는 자식 프로세스 설정.
                    switch(menu) {
                        case 1:
                            // 기본 CFS 실행을 위해 속성을 설정하지 않음.
                            break;
                        case 2:
                            runCFSNice(i);
                            break;
                        case 3:
                            runFIFO();
                            break;
                        case 4:
                            runRR();
                            break;
                    }

                    // 계산 함수 시작.
                    benchFunction();

                    // 종료시간 기록.
                    gettimeofday(&time_end, NULL);
                    printInfo(menu, getpid(),
                        getpriority(which, 0), time_start, t
                        ime_end);
                    double execution_time = (time_end.tv_sec - time_start.tv_sec)
                        + (time_end.tv_usec - time_start.tv_usec)
                        / 1000000.0;

                    // 공유 메모리에 결과 저장.
                    shared_memory[i] = (int)(execution_time * 1000000);
                    shmdt(shared_memory);
                    exit(0);
                }
            }
        }
        ...
    }
}

```

해당 부분에서는 실제 선택된 메뉴에 따라 실행되는 부분이다. 먼저 자식 프로세스를 생성하고, 시작된 시각을 기록한다. 이후 선택된 옵션에 맞는 스케줄러와 이에따른 속성을 생성된 프로세스에 대해 적용한다. 그 후 함수 benchFunction을 실행한다. 해당 함수가 끝나게 된 후 계산이 종료된 시각을 기록하고, 프로세스의 정보, 시작 및 종료시간 그리고 수행시 걸린 시간을 출력한다. 마지막으로 수행시 걸린 시간의 평균을 구하기 위해 공유 메모리에 걸린 시간을 기록한 후 자식 프로세스는 종료된다.

```
// Main 함수.
int main() {
    ...
    while(menu) {
        // 자식 프로세스 종료 대기.
        for(int i = 0; i < 21; i++) {
            wait(NULL);
        }

        // 자식 프로세스의 결과값 합산.
        double sum = 0;
        for(int i = 0; i < 21; i++) {
            sum += shared_memory[i] / 1000000.0;
        }

        ...
    }
}
```

해당 부분에서는 모든 자식 프로세스가 종료되기를 기다린 후 자식 프로세스의 실행시간을 공유 메모리에 접근하여 합산한다.

```
// Main 함수.
int main() {
    ...
    while(menu) {
        // 결과 출력.
        printf("Scheduling Policy: ");
        switch(menu) {
            case 1:
                printf("CFS_DEFAULT | ");
                break;
            case 2:
                printf("CFS_NICE | ");
                break;
            case 3:
                printf("RT_FIFO | ");
                break;
            case 4:
                printf("RT_RR | Time Quantum: %d ms | ", getTimeSlice());
                break;
        }

        printf("Average elapsed time: %.6lf\n", (sum / 21.0));

        ...
    }
}
```

이후 선택한 옵션과 합산된 결과에 따라 결과 및 평균을 출력한다.

```
// Main 함수.
int main() {
    ...
    // 스케줄러 복구.
    int policy = SCHED_OTHER;
    struct sched_param param;
    param.sched_priority = 0;
    sched_setscheduler(0, policy, &param);
}

return 0;
}
```

마지막으로 스케줄러를 기본으로 변경한 후 프로그램은 종료된다.

5. 실행 결과

A. CFS_DEFAULT

PID: 3724	NICE: 0	Start time: 11:13:13.596318	End time: 11:13:28.448849	Elapsed time: 14.852531
PID: 3737	NICE: 0	Start time: 11:13:13.664240	End time: 11:13:28.497540	Elapsed time: 14.833300
PID: 3729	NICE: 0	Start time: 11:13:13.620268	End time: 11:13:28.547532	Elapsed time: 14.927264
PID: 3721	NICE: 0	Start time: 11:13:13.632292	End time: 11:13:28.514989	Elapsed time: 14.882697
PID: 3727	NICE: 0	Start time: 11:13:13.608257	End time: 11:13:28.515598	Elapsed time: 14.907341
PID: 3736	NICE: 0	Start time: 11:13:13.648243	End time: 11:13:28.553412	Elapsed time: 14.905169
PID: 3723	NICE: 0	Start time: 11:13:13.604255	End time: 11:13:28.548031	Elapsed time: 14.943776
PID: 3731	NICE: 0	Start time: 11:13:13.628279	End time: 11:13:28.557917	Elapsed time: 14.929638
PID: 3726	NICE: 0	Start time: 11:13:13.600255	End time: 11:13:28.561592	Elapsed time: 14.961337
PID: 3739	NICE: 0	Start time: 11:13:13.656369	End time: 11:13:28.565202	Elapsed time: 14.908833
PID: 3732	NICE: 0	Start time: 11:13:13.636271	End time: 11:13:28.594699	Elapsed time: 14.958428
PID: 3738	NICE: 0	Start time: 11:13:13.652512	End time: 11:13:28.569350	Elapsed time: 14.916838
PID: 3725	NICE: 0	Start time: 11:13:13.593477	End time: 11:13:28.578130	Elapsed time: 14.984653
PID: 3733	NICE: 0	Start time: 11:13:13.640254	End time: 11:13:28.598447	Elapsed time: 14.958193
PID: 3735	NICE: 0	Start time: 11:13:13.645571	End time: 11:13:28.611344	Elapsed time: 14.965773
PID: 3734	NICE: 0	Start time: 11:13:13.644259	End time: 11:13:28.603379	Elapsed time: 14.959120
PID: 3740	NICE: 0	Start time: 11:13:13.660291	End time: 11:13:28.607122	Elapsed time: 14.946831
PID: 3728	NICE: 0	Start time: 11:13:13.612233	End time: 11:13:28.617037	Elapsed time: 15.004804
PID: 3730	NICE: 0	Start time: 11:13:13.624262	End time: 11:13:28.617738	Elapsed time: 14.993476
PID: 3722	NICE: 0	Start time: 11:13:13.616249	End time: 11:13:28.617824	Elapsed time: 15.001575
PID: 3720	NICE: 0	Start time: 11:13:13.668260	End time: 11:13:28.624617	Elapsed time: 14.956357
Scheduling Policy: CFS_DEFAULT Average elapsed time: 14.937997				

Fig 1. CFS_DEFAULT 결과.

Linux kernel에서는 기본 스케줄러는 CFS이고, Nice 값은 0으로 되어있다. 이 결과는 fork를 한 후 어떠한 설정 없이 실행하면 알 수 있었다.

또한 21개의 자식 프로세스 모두 Nice 값이 동일하여 대부분 동일한 시간이 걸린것으로 나왔다.

B. CFS_NICE

PID: 3758	NICE: -20	Start time: 11:13:35.560193	End time: 11:13:41.636917	Elapsed time: 6.076724
PID: 3756	NICE: -20	Start time: 11:13:35.545102	End time: 11:13:41.729789	Elapsed time: 6.184687
PID: 3757	NICE: -20	Start time: 11:13:35.545289	End time: 11:13:41.737600	Elapsed time: 6.192311
PID: 3759	NICE: -20	Start time: 11:13:35.545597	End time: 11:13:41.747468	Elapsed time: 6.201871
PID: 3761	NICE: -20	Start time: 11:13:35.556246	End time: 11:13:41.757008	Elapsed time: 6.200762
PID: 3760	NICE: -20	Start time: 11:13:35.552230	End time: 11:13:41.763260	Elapsed time: 6.211030
PID: 3755	NICE: -20	Start time: 11:13:35.544377	End time: 11:13:41.764763	Elapsed time: 6.220386
PID: 3752	NICE: 0	Start time: 11:13:35.532235	End time: 11:13:47.911891	Elapsed time: 12.379656
PID: 3753	NICE: 0	Start time: 11:13:35.536223	End time: 11:13:47.951635	Elapsed time: 12.415412
PID: 3754	NICE: 0	Start time: 11:13:35.540241	End time: 11:13:47.954013	Elapsed time: 12.413772
PID: 3751	NICE: 0	Start time: 11:13:35.528238	End time: 11:13:47.972347	Elapsed time: 12.444109
PID: 3749	NICE: 0	Start time: 11:13:35.520238	End time: 11:13:47.974364	Elapsed time: 12.454126
PID: 3750	NICE: 0	Start time: 11:13:35.524225	End time: 11:13:47.971437	Elapsed time: 12.447212
PID: 3748	NICE: 0	Start time: 11:13:35.516225	End time: 11:13:47.964025	Elapsed time: 12.447800
PID: 3746	NICE: 19	Start time: 11:13:35.508210	End time: 11:13:52.635342	Elapsed time: 17.127132
PID: 3745	NICE: 19	Start time: 11:13:35.506386	End time: 11:13:52.656826	Elapsed time: 17.150440
PID: 3743	NICE: 19	Start time: 11:13:35.668241	End time: 11:13:52.659632	Elapsed time: 16.991391
PID: 3741	NICE: 19	Start time: 11:13:35.980235	End time: 11:13:52.655412	Elapsed time: 16.675177
PID: 3744	NICE: 19	Start time: 11:13:35.580243	End time: 11:13:52.661983	Elapsed time: 17.081740
PID: 3747	NICE: 19	Start time: 11:13:35.512247	End time: 11:13:52.669144	Elapsed time: 17.156897
PID: 3742	NICE: 19	Start time: 11:13:35.792213	End time: 11:13:52.670451	Elapsed time: 16.878238
Scheduling Policy: CFS_NICE Average elapsed time: 11.873851				

Fig 2. CFS_NICE 결과.

Linux Kernel에서는 Nice 값이 낮은 값일수록 높은 우선순위를 가지게 된다. 이러한 이유로 인해 가장 낮은 Nice 값(-20)을 가진 프로세스를 가장 늦게 실행하더라도 높은 우선순위로 인해 적은 시간이 걸리고, 가장 빨리 끝나게 되었다.

또한 3개의 구간으로 나누어 실행했을 때 Nice 값이 커질수록 실행이 끝나기까지 걸리는 시간이 점진적으로 늘어나는 것을 알 수 있다.

결론적으로 CFS에서 Nice 값은 프로세스 실행의 우선순위와 실행시간에 영향을 줄 수 있었다.

C. RT_FIFO

PID: 3766	Start time: 11:13:59.203423	End time: 11:14:00.098161	Elapsed time: 0.894738
PID: 3767	Start time: 11:14:00.099300	End time: 11:14:01.128824	Elapsed time: 1.029524
PID: 3769	Start time: 11:14:00.492521	End time: 11:14:02.297115	Elapsed time: 1.804594
PID: 3770	Start time: 11:14:00.496312	End time: 11:14:03.427661	Elapsed time: 2.931349
PID: 3771	Start time: 11:14:00.500283	End time: 11:14:04.489666	Elapsed time: 3.989383
PID: 3772	Start time: 11:14:00.504290	End time: 11:14:05.481994	Elapsed time: 4.977704
PID: 3773	Start time: 11:14:00.508273	End time: 11:14:06.310495	Elapsed time: 5.802222
PID: 3774	Start time: 11:14:00.512294	End time: 11:14:07.199026	Elapsed time: 6.686732
PID: 3775	Start time: 11:14:00.516250	End time: 11:14:07.961760	Elapsed time: 7.445510
PID: 3776	Start time: 11:14:00.520246	End time: 11:14:09.086170	Elapsed time: 8.565924
PID: 3777	Start time: 11:14:00.524267	End time: 11:14:10.081252	Elapsed time: 9.556985
PID: 3778	Start time: 11:14:00.528294	End time: 11:14:10.864994	Elapsed time: 10.336700
PID: 3779	Start time: 11:14:00.532263	End time: 11:14:11.433278	Elapsed time: 10.901015
PID: 3780	Start time: 11:14:00.536278	End time: 11:14:12.308422	Elapsed time: 11.772144
PID: 3781	Start time: 11:14:00.540254	End time: 11:14:13.174671	Elapsed time: 12.634417
PID: 3782	Start time: 11:14:01.444778	End time: 11:14:13.945984	Elapsed time: 12.501206
PID: 3768	Start time: 11:14:01.448275	End time: 11:14:14.711269	Elapsed time: 13.262994
PID: 3765	Start time: 11:14:01.452326	End time: 11:14:15.308244	Elapsed time: 13.855918
PID: 3764	Start time: 11:14:01.456265	End time: 11:14:15.920430	Elapsed time: 14.464165
PID: 3763	Start time: 11:14:01.460275	End time: 11:14:16.453405	Elapsed time: 14.993130
PID: 3762	Start time: 11:14:01.464270	End time: 11:14:17.099461	Elapsed time: 15.635191
Scheduling Policy: RT_FIFO Average elapsed time: 8.763883			

Fig 3. RT_FIFO 결과.

RT FIFO의 경우 후반에 실행된 프로세스일수록 처리에 걸리는 시간이 오래 걸리는것으로 나온다. 이러한 결과가 나오게 되는 이유는, 먼저 도착한 프로세스를 처리하고 다음 프로세스의 작업을 실행시키기 위해 대기하는 시간이 늘어나게 되면서 걸리는 시간이 증가하기 때문이다.

D. RT_RR

PID: 3795	Start time: 11:14:54.453261	End time: 11:14:54.989806	Elapsed time: 0.536545
PID: 3794	Start time: 11:14:54.990542	End time: 11:14:56.000951	Elapsed time: 1.010409
PID: 3798	Start time: 11:14:55.504212	End time: 11:15:11.578892	Elapsed time: 16.074680
PID: 3804	Start time: 11:14:55.536180	End time: 11:15:11.743865	Elapsed time: 16.207685
PID: 3805	Start time: 11:14:55.540186	End time: 11:15:11.753173	Elapsed time: 16.212987
PID: 3797	Start time: 11:14:55.500215	End time: 11:15:11.776948	Elapsed time: 16.276733
PID: 3801	Start time: 11:14:55.520213	End time: 11:15:11.815457	Elapsed time: 16.295244
PID: 3803	Start time: 11:14:55.532181	End time: 11:15:11.843910	Elapsed time: 16.311729
PID: 3796	Start time: 11:14:55.492590	End time: 11:15:11.764055	Elapsed time: 16.271465
PID: 3793	Start time: 11:14:55.496311	End time: 11:15:11.927742	Elapsed time: 16.431431
PID: 3800	Start time: 11:14:55.516219	End time: 11:15:11.951871	Elapsed time: 16.435652
PID: 3802	Start time: 11:14:55.524237	End time: 11:15:11.958342	Elapsed time: 16.434105
PID: 3791	Start time: 11:14:55.528222	End time: 11:15:11.959516	Elapsed time: 16.431294
PID: 3792	Start time: 11:14:55.508199	End time: 11:15:12.034734	Elapsed time: 16.526535
PID: 3799	Start time: 11:14:55.512201	End time: 11:15:12.041664	Elapsed time: 16.529463
PID: 3789	Start time: 11:14:56.464254	End time: 11:15:12.180105	Elapsed time: 15.715851
PID: 3807	Start time: 11:14:56.448259	End time: 11:15:12.192752	Elapsed time: 15.744493
PID: 3808	Start time: 11:14:56.452267	End time: 11:15:12.196020	Elapsed time: 15.743753
PID: 3809	Start time: 11:14:56.456264	End time: 11:15:12.203050	Elapsed time: 15.746786
PID: 3790	Start time: 11:14:56.460276	End time: 11:15:12.207430	Elapsed time: 15.747154
PID: 3806	Start time: 11:14:56.444699	End time: 11:15:12.226982	Elapsed time: 15.782283
Scheduling Policy: RT_RR Time Quantum: 10 ms Average elapsed time: 14.688870			

Fig 4. RT_RR, 10ms의 Time Slice의 결과.

PID: 3822	Start time: 11:15:24.630653	End time: 11:15:25.729928	Elapsed time: 1.099275
PID: 3821	Start time: 11:15:25.731278	End time: 11:15:30.103463	Elapsed time: 4.372185
PID: 3823	Start time: 11:15:26.492467	End time: 11:15:38.166835	Elapsed time: 11.674368
PID: 3820	Start time: 11:15:26.496170	End time: 11:15:38.215188	Elapsed time: 11.719018
PID: 3819	Start time: 11:15:26.500185	End time: 11:15:38.260027	Elapsed time: 11.759842
PID: 3818	Start time: 11:15:26.512178	End time: 11:15:38.572486	Elapsed time: 12.060308
PID: 3826	Start time: 11:15:26.516180	End time: 11:15:38.645309	Elapsed time: 12.129129
PID: 3833	Start time: 11:15:27.452181	End time: 11:15:39.119990	Elapsed time: 11.667809
PID: 3829	Start time: 11:15:26.528182	End time: 11:15:39.567223	Elapsed time: 13.039041
PID: 3830	Start time: 11:15:26.532197	End time: 11:15:39.613473	Elapsed time: 13.081276
PID: 3831	Start time: 11:15:26.536174	End time: 11:15:39.677228	Elapsed time: 13.141054
PID: 3832	Start time: 11:15:26.540213	End time: 11:15:39.704962	Elapsed time: 13.164749
PID: 3824	Start time: 11:15:26.504204	End time: 11:15:39.720881	Elapsed time: 13.216677
PID: 3825	Start time: 11:15:26.508173	End time: 11:15:39.732608	Elapsed time: 13.224435
PID: 3827	Start time: 11:15:26.520183	End time: 11:15:39.751046	Elapsed time: 13.230863
PID: 3828	Start time: 11:15:26.524314	End time: 11:15:39.780477	Elapsed time: 13.256163
PID: 3834	Start time: 11:15:27.444434	End time: 11:15:39.862405	Elapsed time: 12.417971
PID: 3817	Start time: 11:15:27.448187	End time: 11:15:39.894877	Elapsed time: 12.446690
PID: 3835	Start time: 11:15:27.456199	End time: 11:15:39.898011	Elapsed time: 12.441812
PID: 3836	Start time: 11:15:27.460188	End time: 11:15:39.935012	Elapsed time: 12.474824
PID: 3816	Start time: 11:15:27.464214	End time: 11:15:39.978606	Elapsed time: 12.514392
Scheduling Policy: RT_RR Time Quantum: 100 ms Average elapsed time: 11.625328			

Fig 5. RT_RR, 100ms의 Time Slice의 결과.

PID: 3892	Start time: 11:15:56.112649	End time: 11:15:56.814358	Elapsed time: 0.701709
PID: 3891	Start time: 11:15:56.815253	End time: 11:15:57.447625	Elapsed time: 0.632372
PID: 3893	Start time: 11:15:57.452663	End time: 11:15:58.360253	Elapsed time: 0.907590
PID: 3890	Start time: 11:15:57.496317	End time: 11:15:59.090116	Elapsed time: 1.593799
PID: 3894	Start time: 11:15:57.500289	End time: 11:15:59.763552	Elapsed time: 2.263263
PID: 3895	Start time: 11:15:57.504249	End time: 11:16:00.389480	Elapsed time: 2.885231
PID: 3889	Start time: 11:15:57.508236	End time: 11:16:00.977887	Elapsed time: 3.469651
PID: 3896	Start time: 11:15:57.512241	End time: 11:16:01.742900	Elapsed time: 4.230659
PID: 3897	Start time: 11:15:57.524305	End time: 11:16:02.617828	Elapsed time: 5.093523
PID: 3898	Start time: 11:15:57.528315	End time: 11:16:03.189041	Elapsed time: 5.660726
PID: 3888	Start time: 11:15:57.532262	End time: 11:16:03.975459	Elapsed time: 6.443197
PID: 3899	Start time: 11:15:57.536260	End time: 11:16:04.769341	Elapsed time: 7.233081
PID: 3900	Start time: 11:15:57.540267	End time: 11:16:05.321066	Elapsed time: 7.780799
PID: 3901	Start time: 11:15:58.456723	End time: 11:16:05.982839	Elapsed time: 7.526116
PID: 3902	Start time: 11:15:58.460278	End time: 11:16:06.649101	Elapsed time: 8.188823
PID: 3903	Start time: 11:15:58.464284	End time: 11:16:07.601601	Elapsed time: 9.137317
PID: 3904	Start time: 11:15:58.468258	End time: 11:16:08.602343	Elapsed time: 10.134085
PID: 3906	Start time: 11:15:58.476256	End time: 11:16:10.641671	Elapsed time: 12.165415
PID: 3887	Start time: 11:15:58.480249	End time: 11:16:11.579776	Elapsed time: 13.099527
PID: 3886	Start time: 11:15:58.484241	End time: 11:16:12.430457	Elapsed time: 13.946216
PID: 3905	Start time: 11:15:58.472253	End time: 11:16:12.466744	Elapsed time: 13.994491
Scheduling Policy: RT_RR Time Quantum: 1000 ms Average elapsed time: 6.527980			

Fig 6. RT_RR, 1,000ms의 Time Slice의 결과.

PID: 3922	Start time: 11:16:56.157034	End time: 11:16:57.091380	Elapsed time: 0.934346
PID: 3921	Start time: 11:16:57.104869	End time: 11:16:57.996835	Elapsed time: 0.891966
PID: 3920	Start time: 11:16:57.504394	End time: 11:16:58.961465	Elapsed time: 1.457071
PID: 3923	Start time: 11:16:57.508246	End time: 11:16:59.941535	Elapsed time: 2.433289
PID: 3919	Start time: 11:16:57.512272	End time: 11:17:00.816633	Elapsed time: 3.304361
PID: 3924	Start time: 11:16:57.516256	End time: 11:17:01.667033	Elapsed time: 4.150777
PID: 3925	Start time: 11:16:57.520250	End time: 11:17:02.793661	Elapsed time: 5.273411
PID: 3926	Start time: 11:16:57.524244	End time: 11:17:03.724194	Elapsed time: 6.199950
PID: 3918	Start time: 11:16:57.528292	End time: 11:17:04.392418	Elapsed time: 6.864126
PID: 3927	Start time: 11:16:57.532267	End time: 11:17:05.265230	Elapsed time: 7.732963
PID: 3928	Start time: 11:16:57.536289	End time: 11:17:06.100407	Elapsed time: 8.564118
PID: 3929	Start time: 11:16:57.540282	End time: 11:17:06.854524	Elapsed time: 9.314242
PID: 3930	Start time: 11:16:58.456801	End time: 11:17:07.699271	Elapsed time: 9.242470
PID: 3931	Start time: 11:16:58.460291	End time: 11:17:08.407878	Elapsed time: 9.947587
PID: 3932	Start time: 11:16:58.464298	End time: 11:17:09.048043	Elapsed time: 10.583745
PID: 3933	Start time: 11:16:58.468292	End time: 11:17:09.705319	Elapsed time: 11.237027
PID: 3934	Start time: 11:16:58.472285	End time: 11:17:10.260231	Elapsed time: 11.787946
PID: 3935	Start time: 11:16:58.476272	End time: 11:17:11.103911	Elapsed time: 12.627639
PID: 3936	Start time: 11:16:58.480263	End time: 11:17:12.034093	Elapsed time: 13.553830
PID: 3917	Start time: 11:16:58.484247	End time: 11:17:12.974159	Elapsed time: 14.489912
PID: 3916	Start time: 11:16:58.488284	End time: 11:17:13.683091	Elapsed time: 15.194807
Scheduling Policy: RT_RR Time Quantum: 5000 ms Average elapsed time: 7.894552			

Fig 7. RT_RR, 5,000ms의 Time Slice의 결과(추가 실험).

RT_RR에서는 Time Slice를 각각 10, 100, 1,000 ms를 명령어를 통해 설정 후 실험해보았다. 각 실행에서 10 ms의 경우 가장 높은 평균 실행시간을 보여주었는데, 이는 Context Switching이 다른 Time Slice 설정에 비해 자주 일어났기 때문으로 보인다.

또한 Time Slice가 늘어날수록 점점 FIFO로 수렴해 나가는것이 보여 추가 실험을 진행해보았다. Time Slice가 충분히 큰 값(5,000 ms)일 때 FIFO의 평균 실행값과 비교해보았을 때 둘 간의 차가 오차범위 내임을 확인할수 있었다.

결론적으로 RR 스케줄러를 이용할 때 Time Slice가 충분히 큰 값이 된다면 FIFO 스케줄러와 비슷한 효과를 낼 수 있었다.