[COSE341-02] 운영체제

Assignment #2

정보대학 데이터과학과 2021320307 김은진

2023-12-4 (freeday 4일 사용)

1. Development environment

Windows 11 intel core i5 노트북, Virtual Box, Ubuntu 18.04.2 LTS, Linux 4.20.11 Kernel

2. Explanation of CPU scheduling and policies

1. CPU scheduling

cpu 스케줄링은 cpu 사용량을 최대화하고 cpu idle time을 최소화 하기 위한 작업이다. 여러 프로세스가 cpu를 공유하고 있는 상황에서 어떤 프로세스를 어떤 순서로 실행할지를 결정하고, ready queue에 있는 프로세스 중 하나를 선택하여 run queue로 이동시켜 cpu를 할당한다.

Cpu burst는 cpu가 계산을 수행하기 위해 사용되는 시간을 나타내며, I/O burst는 I/O processing을 기다리는 시간을 의미한다. 프로세스는 일반적으로 cpu burst와 I/O burst를 반복하며, 어느 쪽에 더 많은 시간을 사용하는지에 따라 cpu-intensive 또는 I/O-intensive로 분류된다. 시스템이 어느 종류의 프로세스를 많이 실행하는지에 따라 스케줄링 기법의 효율이 결정된다.

Dispatcher는 cpu 스케줄링을 관리하는 구현으로,. Context switching이 발생하여 유저 프로그램에서 프로그램이 재시작될 적절한 위치로 이동한다. Dispatcher가 호출되는 경우는 voluntary yield와 time sharing이 있다. Voluntary yield는 프로세스가 I/O 작업을 위해 CPU를 반환하는 경우이고, time sharing은 정해진 시간 간격으로 주기적으로 CPU가 할당되는 경우이다. Dispatcher가 호출되면 User mode에서 kernel mode로 mode switch를 하고, PCB에 이전 process의 실행상태를 저장한 후 state queue에 넣는다. 그리고 실행될 프로세스(next process)를 결정하여 해당 프로세스의 state를 PCB에서 레지스터로 load한 후 kernel mode에서 user mode로 mode switch한 뒤, 새로운 프로세스의 instriction으로 점프한다. 간략히 말하면, dispatcher는 다음 task를 결정하고 현재 task를 적절한 queue에 넣어주는 작업을 수행한다.

Cpu 스케줄링은 비선점형 스케줄링과 선점형 스케줄링으로 나눌 수 있다. 비선점형 스케줄링에서는 운영체제가 프로세스의 cpu 사용을 조종할 수 없으며 실행중인 프로세스만 cpu를 해제할 수 있다. 선점형 스케줄링에서는 운영체제가 현재 cpu를 사용하고

있는 프로세스를 바꿀 수 있으며, 타이머 기반의 time sharing이 그 예시이다.

2. scheduling policies

스케줄링 policy는 cpu 스케줄링에서 사용되는 알고리즘을 의미하며, 여러 프로세스가 cpu를 공유할 때 어떤 순서로 cpu을 할당 받을지 결정한다.

Scheduling criteria는 cpu 스케줄링 policy를 평가하고 선택하는 데 사용되는 기준이다. 대표적으로 아래의 다섯가지가 있는데, 모든 기준을 이상적으로 만족하는 것은 사실상 불가능하기 때문에 시스템의 목적에 따라 적절한 criteria를 활용한다.

- cpu utilization: 전체 시간 중 cpu가 활용되는 시간의 비율
- waiting time: 프로세스가 시스템에 도착한 후 CPU를 할당받을 때까지 대기하는 시간
- throughput: 주어진 시간 동안 시스템이 처리할 수 있는 프로세스의 수
- turnaround time: 프로세스의 생성부터 종료까지 걸리는 시간
- response time: 프로세스가 생성된 후 처음으로 cpu를 점유하기까지 걸리는 시간

몇 가지 주요 스케줄링 정책에는 아래 네가지가 있다.

1) FCFS(비선점형)

   먼저 도착한 프로세스가 먼저 CPU를 할당받는 가장 간단한 스케줄링 policy이다.

1) SJF(비선점형)

   예상 실행 시간이 가장 짧은 프로세스에게 CPU를 할당하는 방식이다.

2) SRTF(선점형)

   현재 실행 중인 프로세스의 남은 실행 시간보다 더 짧은 실행 시간을 가진 프로세스에게 CPU를 할당하는 방식으로, 새로운 프로세스가 도착할 때 마다 프로세스의 남은 cpu-burst를 확인한다.

3) RR(선점형)

   각 프로세스에 time quantum을 할당하고, 시간이 지나면 다음 프로세스로 넘

어가는 방식이다. 스케줄러가 interrupt로 time slice를 인식하며, 프로세스의 CPU 사용 차례가 끝나면 ready queue의 가장 마지막으로 이동하여 다음 할당을 기다린다.

3. Implementation: modified and written codes with descriptions

1. /arch/x86/entry/syscalls/syscall_64.tbl에 시스템 콜 추가

   System call handler의 이름 정보와 번호를 가지고 있는 파일이다. Linux kernel의 코드 디렉토리에 있는 system call 함수의 주소가 저장된 테이블에 ku_cpu_FCFS, ku_cpu_SJF, ku_cpu_SRTF, ku_cpu_RR 함수와 각각의 번호를 추가하였다.

2. /include/linux/syscalls.h에 시스템 콜 함수 정의

   Syscalls.h는 시스템 콜 함수의 프로토타입을 정의하고 필요한 테이블을 등록하는 파일이다. asmlinkage int sys_ ku_cpu_{scheduling policy명} (char*, int); 네 가지를 추가하였다.

3. /usr/src/linux-4.20.11/kernel/ku_cpu.c에 시스템 콜 함수 구현

   System call handler가 동작하기 위한 실제 소스 코드가 담겨 있는 파일이다. 프로세스의 pid와 job time을 저장할 job_t 구조체를 선언하였고, job_t가 저장될 queue 배열과 queue의 index를 저장하기 위한 rear 변수(0으로 초기화), 현재 실행해야하는 프로세스의 pid를 저장할 now 변수(IDLE, 즉, -1로 초기화), round robin을 구현할 때 활용할 time 변수(0으로 초기화)를 전역 변수로 선언하였다. 구현에 필요한 ku_is_empty, ku_is_new, ku_push, ku_pop, sort_queue_by_job(queue에 있는 프로세스들을 job time이 적은 순서대로 정렬) 함수들을 선언하였다. 모든 handler가 프로세스의 이름과 job time을 인자로 받기 때문에 구현할 때 SYSCALL_DEFINE2 매크로를 활용하였으며, 프로세스가 cpu 점유에 성공하면 0을 반환하고, 실패하면 1을 반환한다.

   1) ku_cpu_FCFS

      현재 실행중인 프로세스가 없으면 인자로 받은 프로세스의 pid를 now에 저장한다. 이후에 현재 실행중인 프로세스가 인자로 받은 프로세스와 같은지를 확인한다. 같다면 프로세스의 job time이 0인지 확인하여 0이 아니라면 커널에 작업중이라는 메시지를 보내고, 0이라면 프로세스의 작업이 끝났다는 얘기이므로 커널에 종료 메세

지를 출력하고 ku_pop을 통해 다음으로 실행할 프로세스의 pid를 now에 저장한다. 만약 큐가 비어있다면 now에 IDLE(-1)을 저장한다. 이후 프로세스에 cpu가 할당되었다는 의미로 0을 return 한다.

인자로 받은 프로세스가 현재 실행중인 프로세스가 아니라면 기존에 큐에 있었던 프로세스인지 확인하고 Push한 후 커널에 요청이 거절되었다는 메세지를 출력한 후 프로세스가 cpu를 할당 받지 못했다는 의미로 1을 return한다.

2) ku_cpu_SJF

대부분의 코드가 FCFS에서와 같고, 현재 실행중인 프로세스와 인자로 받은 프로세스가 다를 때만 차이가 있다. 인자로 받은 프로세스가 큐에 존재하던 프로세스가 아니라면, 즉, 새로 도착한 프로세스라면, 큐에 프로세스를 추가하고 큐에 있는 프로세스들을 job이 적은 순서대로 정렬한다. 이로 인해 실행 중이던 프로세스가 끝났을 때 ku_pop을 통해 job time이 가장 짧은 프로세스의 pid를 now에 저장할 수 있다.

3) ku_cpu_SRTF

현재 실행중인 프로세스가 없다면 인자로 받은 프로세스의 pid를 now에 저장하고, 현재 실행중인 프로세스의 남은 job time을 확인해야 하므로 큐에 프로세스를 push 한다.

이후 현재 실행중인 프로세스가 인자로 받은 프로세스와 같은지 확인한 후 같다면 job이 0인지 확인하여 0이라면 프로세스의 작업이 끝났으므로 커널 메세지를 출력하고 큐에서 해당 프로세스를 pop한다. 큐는 남은 job time이 짧은 순으로 정렬되어 있으므로 now에 queue[0].pid를 저장하는데, 프로세스의 job time을 확인해야 하므로 pop은 하지 않는다. 만약 job이 0이 아니라면 커널에 작업중이라는 메세지를 출력하고, 이후 새로운 프로세스가 도착했을 때 큐를 정렬하기 위해 큐에 저장되어 있는 프로세스의 job을 업데이트해준다.

인자로 받은 프로세스가 현재 실행중인 프로세스가 아니라면 새로운 프로세스가 도착했다는 의미이므로, 큐에 프로세스를 push하고 큐를 job time이 짧은 순서대로 정렬한다. 만약 정렬 후에 큐의 가장 앞에 있는 프로세스가 현재 실행중인 프로세스

와 다르다면 남은 job time이 더 짧은 프로세스가 있다는 뜻이므로, now에 queue[0].pid를 저장하는데, 프로세스의 job time을 확인해야 하므로 pop은 하지 않는다.

SRTF가 선점형 cpu 스케줄링 기법이기 때문에 프로세스의 작업이 끝나지 않았는데도 now에 다른 프로세스의 pid를 지정할 수 있도록 코드를 작성하였다.

4) ku_cpu_RR

현재 실행중인 프로세스가 없다면 인자로 받은 프로세스의 pid를 now에 저장한다. 만약 현재 실행중인 프로세스가 인자로 받은 프로세스라면 job time을 확인하고 job이 0이라면 프로세스의 작업이 끝났으므로 커널에 메세지를 출력 후 time을 0으로 초기화하고 큐에서 pop한 프로세스의 pid를 now에 저장한다. 만약 job이 0이 아니라면 작업중이라는 커널 메세지를 출력하고, 만약 프로세스가 cpu를 time slice만큼 점유했다면 time을 0으로 초기화한 후 pop을 통해 다음으로 실행될 프로세스의 pid를 now에 저장하고, 차례가 끝났다는 커널 메세지를 출력한 후 종료한다. time은 해당 프로세스가 시스템 콜을 호출했을 때만 증가해야 하므로 time이 time slice와 다를 때만 time++가 실행되도록 코드를 작성하였다. RR이 선점형 cpu 스케줄링 기법이기 때문에 프로세스의 작업이 끝나지 않았는데도 now에 다른 프로세스의 pid를 지정할 수 있도록 코드를 작성하였다.

현재 실행중인 프로세스와 인자로 받은 프로세스가 다르다면 원래 큐에 있던 프로세스인지 확인 후 큐에 push한다.

4. /usr/src/linux-4.20.11/kernel/Makefile에 object file 추가

소스코드를 컴파일 할 때 추가한 scheduler도 컴파일 되도록 oby-j 부분에 ku_cpu.o를 추가하였다.

5. /usr/src/linux-4.20.11/p_FCFS.c, p_SJF.c, p_SRTF.c, p_RR.c

추가한 시스템 콜을 호출하는 user process 파일이다. 각각 파일명에 있는 스케줄링 policy가 구현된 시스템콜을 호출하고, 이외의 코드는 전부 같다. 인자로 job time, delay time, 프로세스명을 입력 받아 저장하고, wait을 0으로 선언하였다. Sleep 함수를 통해

delay를 구현하였고, 이후 CPU를 job time만큼 사용하겠다는 메세지를 출력하게 하여 프로세스가 도착했음을 표시하였다. 시스템콜이 0.1초마다 호출되도록 설정하였기 때문에 job에 10을 곱하여 0.1s단위(1초이면 job=10)로 바꿔주었다. 이후 파일에 따라 알맞은 시스템콜을 호출하여 프로세스가 cpu 점유에 성공했다는 반환 값이 올 때만 job을 감소시키고 점유에 실패하였을 때는 wait을 늘리는 작업을 job이 0이 될 때까지 반복한다. usleep(100000)을 통해 앞서 업급했듯이 0.1초의 delay를 주어 시스템콜의 호출 시간을 조절하였다. job이 0임을 시스템 콜에 전달하여야 다음 프로세스를 실행해야 한다는 것을 알 수 있기 때문에 반복문 밖에서 시스템 콜을 한 번 더 호출하여 cpu점유를 점유하지 않도록 한 후 최종 wait time을 출력하고 종료한다.

4. Experiment results and analysis with screenshots of execution results (logs)

Job time이 각각 7, 5, 3이고 delay가 0, 1, 2인 프로세스 A, B, C를 각 scheduling policy에서 실행한 결과는 1 ~ 4와 같다.

1. FCFS

```
eunjin@eunjin-VirtualBox:/usr/src/linux-4.20.11$ ./run_FCFS
eunjin@eunjin-VirtualBox:/usr/src/linux-4.20.11$
Process A : I will use CPU by 7s.

Process B : I will use CPU by 5s.

Process C : I will use CPU by 3s.

Process A : Finish! My total wait time is 0s.

Process B : Finish! My total wait time is 6s.

Process C : Finish! My total wait time is 10s.
```

```
[10762.842986] Working: A
[10762.894991] Working Denied: C
[10762.916198] Working Denied: B
[10762.948284] Working: A
[10762.996505] Working Denied: C
[10763.016319] Working Denied: B
[10763.052302] Working: A
[10763.103566] Working Denied: C
[10763.116400] Working Denied: B
[10763.156908] Working: A
[10763.203706] Working Denied: C
[10763.216536] Working Denied: B
[10763.263595] Working: A
[10763.306901] Working Denied: C
[10763.317675] Working Denied: B
[10763.370872] Working: A
[10763.409528] Working Denied: C
[10763.418754] Working Denied: B
[10763.479079] Working: A
[10763.518436] Working Denied: C
[10763.519218] Working Denied: B
[10763.582223] Working: A
[10763.618563] Working Denied: C
[10763.619880] Working Denied: B
[10767.115661] Working: A
[10767.167806] Working Denied: C
[10767.184221] Working Denied: B
[10767.215799] Process Finish: A
[10767.267947] Working Denied: C
[10767.284386] Working: B
[10767.371860] Working Denied: C
[10767.384472] Working: B
[10767.476220] Working Denied: C
[10767.484805] Working: B
[10767.580482] Working Denied: C
[10767.585621] Working: B
[10767.683901] Working Denied: C
[10767.691872] Working: B
[10767.783926] Working Denied: C
[10767.792012] Working: B
[10767.884021] Working Denied: C
[10767.892052] Working: B
[10767.984080] Working Denied: C
[10767.993020] Working: B
[10768.084569] Working Denied: C
[10768.099945] Working: B
[10768.184638] Working Denied: C
[10768.200147] Working: B
[10773.235312] Working: C
[10773.335346] Working: C
[10773.435538] Working: C
[10773.535613] Working: C
[10773.636157] Working: C
[10773.736699] Working: C
[10773.839759] Working: C
[10773.940354] Working: C
[10774.041075] Working: C
[10774.141439] Working: C
[10774.241693] Working: C
[10774.341801] Working: C
[10774.441835] Working: C
[10774.543065] Working: C
[10774.643256] Working: C
[10774.744923] Working: C
[10774.845864] Working: C
[10774.947162] Working: C
[10775.047559] Working: C
[10775.147969] Working: C
[10775.248285] Working: C
[10775.348332] Working: C
[10775.448384] Process Finish: C
```

Average wait time = (0 + 6 + 10) / 3 = 5.33

2. SJF

```
eunjin@eunjin-VirtualBox:/usr/src/linux-4.20.11$ ./run_SJF
eunjin@eunjin-VirtualBox:/usr/src/linux-4.20.11$
Process A : I will use CPU by 7s.

Process B : I will use CPU by 5s.

Process C : I will use CPU by 3s.

Process A : Finish! My total wait time is 0s.

Process C : Finish! My total wait time is 5s.

Process B : Finish! My total wait time is 9s.
```

```
[11261.646407] Working Denied: B [11265.203501] Working Denied: B [11271.629059] Working: B
[11261.718771] Working Denied: C [11265.277949] Working Denied: C [11271.730444] Working: B
[11261.746349] Working: A        [11265.303527] Working: A        [11271.835671] Working: B
[11261.746515] Working Denied: B [11265.303541] Working Denied: B [11271.938139] Working: B
[11261.819229] Working Denied: C [11265.378756] Working Denied: C [11272.038955] Working: B
[11261.852091] Working: A        [11265.405162] Working Denied: B [11272.139538] Working: B
[11261.852099] Working Denied: B [11265.405173] Working: A        [11272.243861] Working: B
[11261.919273] Working Denied: C [11265.480980] Working Denied: C [11272.347298] Working: B
[11261.953437] Working: A        [11265.511472] Working: A        [11272.448034] Working: B
[11261.953446] Working Denied: B [11265.511480] Working Denied: B [11272.548182] Working: B
[11262.019305] Working Denied: C [11265.581010] Working Denied: C [11272.648238] Working: B
[11262.053538] Working: A        [11265.612379] Process Finish: A [11272.748292] Working: B
[11262.053549] Working Denied: B [11265.612708] Working Denied: B [11272.850149] Working: B
[11262.119483] Working Denied: C [11265.681079] Working: C        [11272.950202] Working: B
[11262.153928] Working: A        [11265.712815] Working Denied: B [11273.050257] Working: B
[11262.153938] Working Denied: B [11265.781479] Working: C        [11273.150306] Working: B
[11262.220507] Working Denied: C [11265.813038] Working Denied: B [11273.250467] Working: B
[11262.255016] Working: A        [11265.881500] Working: C        [11273.350515] Working: B
[11262.255027] Working Denied: B [11265.913103] Working Denied: B [11273.453016] Working: B
[11262.325603] Working Denied: C [11265.981521] Working: C        [11273.554840] Working: B
[11262.355561] Working: A        [11266.013126] Working Denied: B [11273.663279] Working: B
[11262.355569] Working Denied: B [11266.081542] Working: C        [11273.768021] Working: B
[11262.431347] Working Denied: C [11266.113164] Working Denied: B [11273.869898] Process Finish: B
[11262.461852] Working: A        [11266.181568] Working: C
```

Average wait time = (0 + 5 + 9) / 3 = 4.66

3. SRTF

```
eunjin@eunjin-VirtualBox:/usr/src/linux-4.20.11$ ./run_SRTF
eunjin@eunjin-VirtualBox:/usr/src/linux-4.20.11$
Process A : I will use CPU by 7s.

Process B : I will use CPU by 5s.

Process C : I will use CPU by 3s.

Process C : Finish! My total wait time is 0s.

Process B : Finish! My total wait time is 3s.

Process A : Finish! My total wait time is 8s.
```

```
[11491.842859] Working: B        [11495.243437] Working Denied: A [11503.589791] Working: A
[11491.873042] Working Denied: A [11495.292805] Working: C        [11503.689822] Working: A
[11491.942965] Working: B        [11495.309223] Working Denied: B [11503.789959] Working: A
[11491.973076] Working Denied: A [11495.343482] Working Denied: A [11503.890000] Working: A
[11492.043155] Working: B        [11495.393403] Working: C        [11503.990328] Working: A
[11492.078748] Working Denied: A [11495.409241] Working Denied: B [11504.090353] Working: A
[11492.143551] Working: B        [11495.444548] Working Denied: A [11504.191342] Working: A
[11492.186433] Working Denied: A [11495.493587] Process Finish: C [11504.291393] Working: A
[11492.243633] Working: B        [11495.509728] Working: B        [11504.391431] Working: A
[11492.286616] Working Denied: A [11495.558065] Working Denied: A [11504.491947] Working: A
[11492.340734] Working Denied: C [11495.609788] Working: B        [11504.591973] Working: A
[11492.343694] Working Denied: B [11495.658154] Working Denied: A [11504.694820] Working: A
[11492.389170] Working Denied: A [11495.710639] Working: B        [11504.795123] Working: A
[11492.440863] Working: C        [11495.758589] Working Denied: A [11504.904530] Working: A
[11492.443751] Working Denied: B [11495.810663] Working: B        [11505.005461] Working: A
[11492.489207] Working Denied: A [11495.859142] Working Denied: A [11505.105595] Working: A
[11492.540948] Working: C        [11495.912793] Working: B        [11505.218041] Working: A
[11492.543853] Working Denied: B [11495.959172] Working Denied: A [11505.318423] Working: A
[11492.589244] Working Denied: A [11496.015360] Working: B        [11505.425625] Working: A
[11492.640989] Working: C        [11496.067672] Working Denied: A [11505.527056] Working: A
[11492.650083] Working Denied: B [11496.117134] Working: B        [11505.627102] Working: A
[11492.691575] Working Denied: A [11496.170679] Working Denied: A [11505.727436] Working: A
[11492.741079] Working: C        [11496.221872] Working: B        [11505.827630] Process Finish: A
[11492.750623] Working Denied: B [11496.270722] Working Denied: A
```

Average wait time = (0 + 3 + 8) / 3 = 3.66

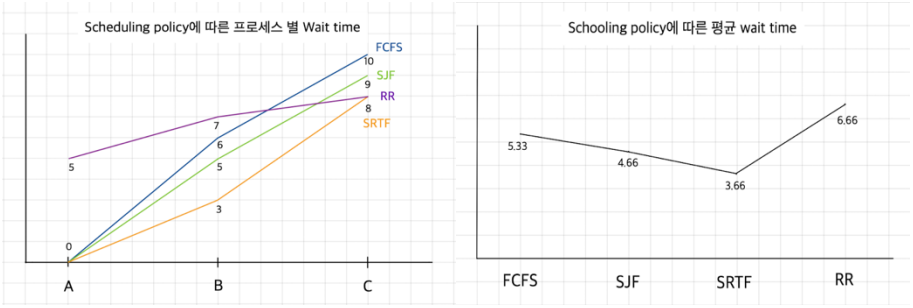
4. RR

```
eunjin@eunjin-VirtualBox:/usr/src/linux-4.20.11$ ./run_RR
eunjin@eunjin-VirtualBox:/usr/src/linux-4.20.11$
Process A : I will use CPU by 7s.

Process B : I will use CPU by 5s.

Process C : I will use CPU by 3s.

Process C : Finish! My total wait time is 5s.

Process B : Finish! My total wait time is 7s.

Process A : Finish! My total wait time is 8s.
```

```
[11769.642457] Working: B       [11777.326690] Working: A       [11780.389890] Working: B
[11769.742482] Working: B       [11777.326691] —>Turn Over: A   [11780.495682] Working: B
[11769.842514] Working: B       [11777.425550] Working: C       [11780.596613] Working: B
[11769.942537] Working: B       [11777.526551] Working: C       [11780.701104] Working: B
[11769.942538] —>Turn Over: B   [11777.632934] Working: C       [11780.807585] Working: B
[11769.959681] Working: A       [11777.732976] Working: C       [11780.912175] Process Finish: B
[11770.060724] Working: A       [11777.833006] Working: C       [11781.020122] Working: A
[11770.161045] Working: A       [11777.934940] Working: C       [11781.121068] Working: A
[11770.261197] Working: A       [11778.035557] Working: C       [11781.225871] Working: A
[11770.361230] Working: A       [11778.135630] Working: C       [11781.325903] Working: A
[11770.461345] Working: A       [11778.243261] Process Finish: C [11781.425927] Working: A
[11770.561435] Working: A       [11778.252252] Working: B       [11781.528572] Working: A
[11770.661555] Working: A       [11778.357651] Working: B       [11781.628846] Working: A
[11770.761652] Working: A       [11778.460396] Working: B       [11781.729061] Working: A
[11770.861734] Working: A       [11778.563257] Working: B       [11781.829096] Working: A
[11770.961876] Working: A       [11778.663348] Working: B       [11781.929133] Working: A
[11770.961877] —>Turn Over: A   [11778.764157] Working: B       [11782.029171] Working: A
[11771.033896] Working: C       [11778.864427] Working: B       [11782.029172] —>Turn Over: A
[11771.133987] Working: C       [11778.964688] Working: B       [11782.130025] Working: A
[11771.234089] Working: C       [11779.064898] Working: B       [11782.230314] Working: A
[11771.334108] Working: C       [11779.164979] Working: B       [11782.330609] Working: A
[11771.434142] Working: C       [11779.265708] Working: B       [11782.435203] Working: A
[11771.534203] Working: C       [11779.265709] —>Turn Over: B   [11782.537101] Process Finish: A
[11771.635917] Working: C       [11779.265719] Working: A
```

Average wait time = (5 + 7 + 8) / 3 = 6.66

5. analysis



그래프에서 알 수 있듯이 wait time은 RR, FCFS, SJF, SRTF 순으로 높게 나타났다. job time이 긴 프로세스가 job time이 짧은 프로세스보다 먼저 도착하는 상황이기 때문에 FCFS에 비해 SJF의 wait time이 짧고, 비선점형이기 때문에 job time이 더 짧은 프로세스가 도착해도 실행할 프로세스를 바꾸지 못하는 SJF에 비해 선점형이 때문에 실행중인 프로세스를 바꿀 수 있는 SRTF의 wait time이 더 짧음을 확인할 수 있다. RR에서는 wait time은 비교적 높게 나타났지만 프로세스 간의 wait time 차이가 작다는 것을 보면 다른 스케줄링 policy들에 비해 공평한 cpu 할당이 이루어지고 있음을 알 수 있다.

6. Characteristics

구현 결과에서 알 수 있는 내용을 포함하여 scheduling policy의 특징을 정리하면 다음과 같다.

1) FCFS(비선점형)

구현이 간단하고 이해하기 쉬우며 평균 대기 시간이나 대기 시간의 변동이 다른 알고리즘에 비해 적지만, 작업시간이 긴 프로세스 뒤에 짧은 프로세스가 실행되는 경우처럼 프로세스의 도착 순서에 따라 평균 waiting time 이나 response time이 길어질 수 있다.

2) SJF(비선점형)

평균 waiting time 측면에서는 효율적이지만, 실행 시간을 정확하게 예측하기 어려워 실제로 구현이 어려울 수 있고, 작업의 길이를 사전에 알기 어려울 때 starvation(기아) 문제가 발생할 수 있다.

3) SRTF(선점형)

평균 waiting time 측면에서는 효율적이지만, 선점형 알고리즘이기 때문에 Context Switching 오버헤드가 발생하고, 프로세스의 실행 시간을 정확하게 예측하기 어려워 이로 인한 성능 저하가 발생할 수 있다. 또한, 실행 시간이 긴 프로세스가 계속해서 짧은 프로세스에게 밀려나 CPU를 할당받지 못하는 starvation 문제가 발생할 수 있어서 이를 방지하기 위해 aging 등의 해결 기법이 필요할 수 있다.

4) RR(선점형)

각 작업에게 동등한 기회를 부여하기 때문에 공정한 스케줄링이 이루어지고, 선점형 알고리즘이지만 일정 시간 동안 각 작업에게 CPU를 할당하기 때문에 starvation 문제가 발생하지 않는다는 장점이 있다. 하지만 평균 대기 시간이 상대적으로 길 수 있고, job time이 긴 경우, 빈번한 context switching으로 오버헤드가 발생할 수 있다. 또한, time slice가 context switch 시간보다 커야한다는 제약이 있으며, Time slice가 크면 FCFS와 유사하게 동작하고, time slice가 작으

면 responsiveness가 좋지만, performance가 낮아지기 때문에 작업의 특성에 따라 time slice를 적절하게 설정해야 한다는 어려움이 있다.

5. Problems encountered during the project and solutions to them

1. 초기 /usr/src/linux-4.20.11/kernel/ku_cpu.c 컴파일 오류

   1) function declaration isn't a prototype error

      인자를 받지 않는 ku_is_empty 함수를 정의할 때 'ku_is_empty()'와 같이 괄호 안에 아무 것도 적지 않아 에러가 발생했다. 'ku_is_empty(void)'로 수정하여 해결하였다.

   2) for loop 내부에서 변수 초기 선언

      C90에서는 'for(int i=0 ; i<rear ; i++)'처럼 for loop 안에서 변수를 초기 선언할 수 없기 때문에 for loop 밖에서 i, j를 미리 선언하도록 수정하였다.

2. ./run_RR 실행했을 때 잘못된 결과 출력

   round robin scheduling policy에서 세 개의 process를 실행한 결과 waiting time이 예상보다 짧게 나오는 결과가 출력되었다. 기존에 time++;를 실행한 후 time==TIMESLICE인지 확인하도록 코드를 작성하여서 프로세스가 이론상 cpu를 점유해야하는 시간보다 1초 덜 점유하게 되어서 발생하는 것이었고, time==TIMESLICE인지 확인 후 증가시키는 것으로 수정하여 해결하였다.