운영체제

과제2

양승민 교수님

컴퓨터학부
공통반
20132375
송은교

1. 소개

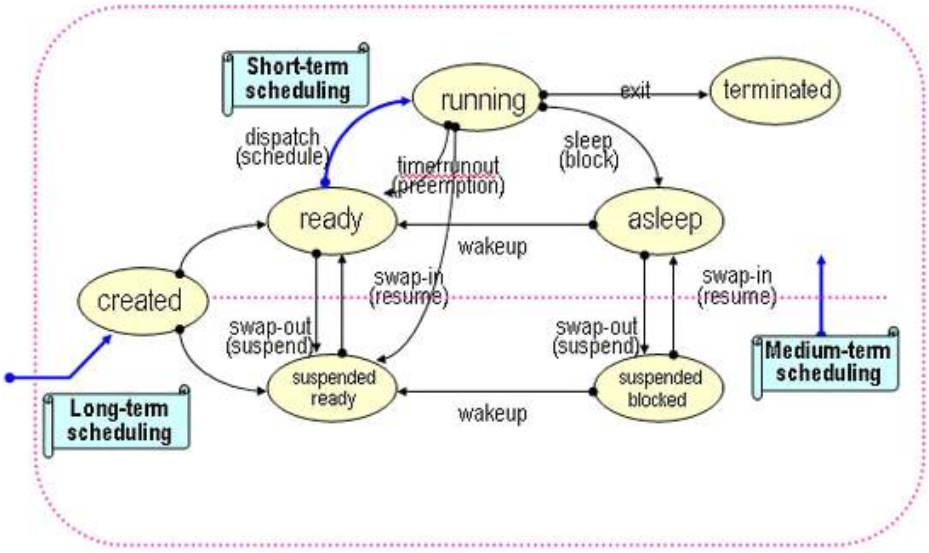
텍스트 형식의 프로세스 스케줄링 목록 입력 파일을 프로그램의 첫 번째 인수로 받아들여 프로세스를 생성하고, SJF, SRT, RR, PR의 네가지 방식의 스케줄링 알고리즘으로 프로세스를 실행하는 프로그램이다. data1. txt 파일에는 한 행에 스케줄링 할 한개의 프로세스 스케줄링 정보가 기술되어 있어, 한 행 씩 읽어들이며,
각 행은 "id arrive-time service-time priority" 형식으로 되어있다. 이 목록 정보를 기반으로 4가지 스케줄링 알고리즘에 대한 시뮬레이션 결과를 stdout 장치로출력해야하며, 결과는 간트 챠트(Gantt Chart) 형태로 출력한다.
본 과제는 4가지 스케줄링 알고리즘 중, PR(Priority Scheduling(preempive))알고리즘을 추가로 구현한다.

2. 관련 연구

1) 스케줄링

스케줄링이란 다중 프로그래밍을 가능하게 하는 운영체제의 동작 기법이다. 운영체제는 프로세스들에게 CPU 등의 자원을 적절히 배정함으로써 시스템의 성능을 개선할 수 있다.
유형으로는 크게 3가지로 나뉜다.

1. 장기스케줄링(Long-term scheduling)
2. 중기 스케줄링 (Mid-term scheduling)
3. 단기 스케줄링(Short-term scheduling)으로 나뉜다.



CPU 스케줄링의 결정 시점은 다음과 같은 프로세스 상태 변화가 있을 때이다.

1. 수행 -> 대기
2. 수행 -> 준비
3. 대기 -> 준비
4. 수행 -> 종료

스케줄링 적용 시점에 따라 비선점형(Non-preemptive)과 선점형(Preemptive)으로 나뉠 수 있다.

<비선점 프로세스 스케줄링>
FCFS 스케줄링(First Come First Served Scheduling)
SJF 스케줄링(Shortest Job First Scheduling)
HRRN 스케줄링(Highest Response Ratio Next Scheduling)

<선점 프로세스 스케줄링>
RR 스케줄링(Round Robin Scheduling)
SRTF 스케줄링(Shortest Remaining-Time First Scheduling)
다단계 큐 스케줄링(Multilevel Queue Scheduling)
다단계 피드백 큐 스케줄링(Multilevel Feedback Queue Scheduling)
RM 스케줄링(Rate Monotonic Scheduling)
EDF 스케줄링(Earliest Deadline First Scheduling)

비선점형은 위의 결정시점 중 1번과 4번의 상황에서만 수행 되며, 선점형은 1번 부터 4 번까지 모든 상황에서 수행된다. 비선점형 스케줄링(Non-preemptive Scheduling)은 어떤 프로세스가 CPU를 할당 받으면 그 프로세스가 종료되거나 입출력요구가 발생하여 자발적으로 중지될 때까지 계속 실행되도록 보장한다. 순서대로 처리되는 공정성이 있고 다음에 처리해 야할 프로세스와 관계없이 응답 시간을 예상할 수 있으며 선점방식보다 스케줄러 호출 빈도 가 낮고 문맥 교환에 의한 오버헤드가 적다. 일괄 처리 시스템에 적합하며, CPU 사용 시간 이 긴 하나의 프로세스가 CPU 사용시간이 짧은 여러 프로세스를 오랫동안 대기 시킬 수 있 으므로, 처리율이 떨어질 수 있다는 단점이 있다.

반면, 선점형 스케줄링(Preemptive Scheduling)는 어떤 프로세스가 CPU를 할당받아 실행 중에 있어도 다른 프로세스가 실행 중인 프로세스를 중지하고 CPU를 강제로 점유할 수 있다. 모든 프로세스에게 CPU 사용 시간을 동일하게 부여할 수 있다. 빠른 응답시간을 요구하는 대화식 시분할 시스템에 적합하며 긴급한 프로세서를 제어할 수 있다. '운영 체제가 프로세 서 자원을 선점'하고 있다가 각 프로세스의 요청이 있을 때 특정 요건들을 기준으로 자원을 배분하는 방식이다.

또한 프로세스의 우선순위 변동 여부에 따라 정적 스케줄링과 동적 스케줄링으로 구분할 수 있다.

정적 스케줄링(Static Scheduling)
프로세스에 부여된 우선순위가 바뀌지 않는다. 고정 우선순위 스케줄링이라고도 한다.

동적 스케줄링(Dynamic Scheduling)
스케줄링 과 정에서 우선순위를 변동시킨다. 유동우선순위 스케줄링이라고도 한다.

2) 스케줄링 알고리즘

1. SJF

최단 작업 우선 스케줄링(Shortest Job First Scheduling)

평균 대기 시간을 최소화하기 위해 CPU 점유 시간이 가장 짧은 프로세스에 CPU를 먼저 할당하는 방식의 CPU 스케줄링 알고리즘으로 평균 대기시간을 최소로 만드는 걸 최적으로 두고 있는 알고리즘이다.
요구 시간이 짧은 프로세스가 긴 프로세스에게 항상 양보되어야 하기 때문에 기아 상태가 발생할 수 있으며, 대기 상태에 있는 프로세스의 요구시간에 대한 정확한 자료를 얻기 어렵다는 문제점이 있다.
단기 스케줄링 보다는 장기 스케줄링에 유리하다.

2. SRF

최소 잔류 시간 우선 스케줄링 (shortest remaining time)

SJF 스케줄링을 비선점 형태에서 선점 형태로 수정한 스케줄링 알고리즘으로, 현재 작업 중인 프로세스를 중단시키고 새로 들어온 프로세스의 처리를 시작하는 방식이다.
SRT 스케줄링 ,SRTF 스케줄링 이라고도 부른다.

3. RR

라운드 로빈 스케줄링(Round Robin Scheduling, RR)

시분할 시스템을 위해 설계된 선 점형 스케줄링의 하나로서, 프로세스 사이에 우선순위를 두지 않고,
순서대로 시간단위 (Time Quantum)로 CPU를 할당하는 방식의 CPU 스케줄링 알고리즘이다.
보통 시간 단위는 10 ms ~ 100 ms 정도이다.
시간 단위동안 수행한 프로세스는 준비 큐의 끝으로 밀려나게 된다.
문맥 전환의 오버헤드가 큰 반면, 응답시간이 짧아지는 장점이 있어 실시간 시스템에 유리하다.

4. PR

우선순위 스케줄링(Priority Scheduling)

PR 스케줄링은 각각의 프로세스마다 우선순위를 부여한 후에 우선순위가 가장 높은 프로세스를 먼저 실행시키는 알고리즘이다. Nonpreemptive / preemptive 둘다 가능하다.
Priority Scheduling에는 우선순위가 계속 높은 프로세스만 Ready Queue에 들어오게 되어 낮은 것은 무한히 처리가 되지 않는 문제인 기아현상이 발생한다. 이 문제를 해결하기 위한 방법으로 한 프로세스가 기다리는 시간에 비례하여 우선순위를 높여주는 Aging 기법이 있다.

3) API 함수 조사

1. isupper()

check_valid_id(char*) 함수에 있는 함수로, process id를 판별하기 위해 사용된 함수.
인수로 받은 문자가 대문자인지 판별하는 함수이다.

2. isdigit()

check_valid_id(char*) 함수에 있는 함수로, process id를 판별하기 위해 사용된 함수.
인수로 받은 문자가 숫자 문자인 (‘0’~’9’)지를 판별한다.

3. strchr()

read_config(char *)함수에 있는 함수로 문자열에 공백이 있는지 검사하는데 쓰이는 함수.
문자열에서 임의의 문자가 처음으로 발견된 위치를 구한다.
발견된 바이트의 위치를 포인터로 반환한다.

4. long strtol(const char *restrict str, char **endptr, int base)

숫자 문자열을 long형 숫자로 변환.
Atoi()나 atoll()과는 달리 변환 하려는 진수를 선택 가능하다.
숫자가 아닌 문자를 만나면 그 포인터 위치를 구한다.

char *str ： 정수로 변환할 문자열
char **endptr : 숫자로 변경하지 못하는 문자열의 시작 위치.
int base : 문자열이 가지고 있는 숫자의 진수(2~32)

3. 추가 기능 구현

`static void simulate (int sched)` 함수 내부에 존재하는 switch문에

```
      case SCHED_PR://추가된 PR 스케줄링 알고리즘
{
            int i;
            int temp_pr; //temporary priority를 저장할 변수

            temp_pr = PRIORITY_MAX+1;
            for(i = 0;i < queue_len;i++)
            {
                if(queue[i] -> priority < temp_pr)
                {
                    process = queue[i];
                    temp_pr = process -> priority;
                }
            }
}
```

위와 같은 코드를 넣어주었다.
최소 priority 를 저장할 변수 temp_pr를 선언하였고, 그 변수의 초기화 값으로는 PRIORITY_MAX값에 1을 더해주었다.

4. 실행 화면

```
eunkyo@eunkyo-VirtualBox:~/다운로드$ ./sched data1.txt
duplicate process id 'P3' in line 11, ignored
invalid arrive-time '-1' in line 14, ignored
invalid arrive-time '31' in line 15, ignored
invalid arrive-time '0' in line 18, ignored
invalid arrive-time '0' in line 19, ignored
invalid priority '0' in line 22, ignored
invalid priority '11' in line 23, ignored
invalid priority '-1' in line 24, ignored

[SJF]
P1 ***
P2    ******
P3           ****
P4               *****
P5          **
CPU TIME: 20
AVERAGE TURNAROUND TIME: 7.60
AVERAGE WAITING TIME: 3.60

[SRT]
P1 ***
P2    *      *****
P3     ****
P4               *****
P5         **
CPU TIME: 20
AVERAGE TURNAROUND TIME: 7.20
AVERAGE WAITING TIME: 3.20

[RR]
P1 ***
P2    ** *  *   *   *
P3      * *  *   *
P4         *   *   * **
P5            *   *
CPU TIME: 20
AVERAGE TURNAROUND TIME: 10.40
AVERAGE WAITING TIME: 6.40

[PR]
P1 ***
P2    *      *****
P3     ****
P4               *****
P5         **
CPU TIME: 20
AVERAGE TURNAROUND TIME: 7.20
AVERAGE WAITING TIME: 3.20
```