1. 소개

본 과제는 linux 환경에서 Shortest-Job-First Scheduling, Shortest-Reamining Time-First Scheduling, Round-Robin Scheduling, Priority Scheduling 이 4개의 운영체제 스케줄러 알고리즘을 시뮬레이션 하는 프로그램을 구현하는 것을 목표로 하고 있습니다.
우선순위을 기준으로 하여 스케쥴링 되는 알고리즘인 Priority Scheduling을 추가 구현하는데 성공하였습니다.

본 과제의 구현 및 실행 / 테스트는 linux 운영체제 내에서 실시하였습니다.

2. 관련 연구

2.1 스케줄링 알고리즘

SJF (비선점형)	Shortest Job First Scheduling은 이름대로 평균 대기 시간을 최소화하기 위해 CPU 점유 시간이 가장 짧은 프로세스에 CPU를 먼저 할당하는 방식의 스케줄링 알고리즘이다. 즉, 본 과제에서 도착한 프로세스 중에서 3번째 인자이며 프로세스 서비스 시간을 명시하는 service-time이 가장 짧은 프로세스에 먼저 CPU를 할당한다는 것이다.
SRT (preemptive)	Shortest Remaining Time Scheduling은 SJF를 선점형으로 바꾼 알고리즘이다. 위 알고리즘은 진행 중인 프로세스가 있어도 중단시키고 최단잔여시간 프로세스에 우선권을 부여한다. 즉, 현재 진행 중인 프로세스의 남은 시간과 도착해있는 프로세스들의 실행시간을 비교하여 진행 중인 프로세스의 남은 시간이 가장 적다면 진행 중인 프로세스에 할당된 CPU를 남겨두고 완료한다. 만약 도착해있는 프로세스 중 실행시간이 더 짧은 것이 존재한다면 CPU를 해당 프로세스에 할당한다.
RR (preemptive)	Round Robin Scheduling은 프로세스들 사이에 우선순위를 두지 않고, 순서대로 시간단위로 CPU를 할당하는 방식이다. 도착하여 큐에 존재하는 모든 프로세스들에 CPU를 순서대로 번갈아 가며 CPU를 할당하는 방식이다. 보통 실시간 시스템에 유리하다.
PR (preemptive)	Priority Scheduling은 프로세스에 우선순위를 부여하여 우선순위가 높은 프로세스에 CPU를 우선적으로 할당하는 방식이다. 즉, 프로세스가 실행되고 있을 때 도착한 프로세스의 우선순위가 더 높다면 진행중이던 프로세스는 중지하고 도착한 프로세스에 CPU를 할당한다.

2.2 API 함수

함수	내용	반환형
strlen( ) [string.h]	string의 길이 판별	int
isspace( ) [ctype.h]	문자가 공백 문자인지를 판별	int [0 or 1]
memmove( ) [string.h]	메모리 영역을 복사	void *
isupper( ) [ctype.h]	인수로 받은 문자가 대문자인지 판별	int [0 or 1]
isdigit( ) [ctype.h]	인수로 받은 문자가 숫자인지 판별	int [0 or 1]
strcmp( ) [string.h]	2개의 문자열을 비교	int [0 or 1]
fopen( ) [stdio.h]	파일을 열고 파일 포인터 반환	FILE*
fgets( ) [stdio.h]	2번째 인자로 지정한 개수의 문자를 입력 받을 때 까지 or 개행 문자나 EOF에 도달할 때 까지 입력 받아서 문자열로 저장	char*
memset( ) [string.h]	동적으로 할당 받은 메모리를 특정 값 (두 번째 인자)으로 초기화	void*
strchr( ) [string.h]	문자열에서 특정한 문자가 가장 먼저 나타나는 곳의 위치를 찾고 해당 위치를 가리키는 포인터를 리턴	char*
strcpy( ) [string.h]	문자열 복사	char*
strtol( ) [stdlib.h]	숫자 문자열을 long형 숫자로 변환 변환하려는 진수 선택 가능 숫자 문자가 아닌 문자를 만나면 포인터 위치를 구해줌	long
fclose() [stdio.h]	인자로 지정한 스트림에 해당하는 파일 닫기	성공 : 0 실패 : EOF
printf( ) [stdio.h]	문자열에 지정되어 있는 형태로 출력	int
putchar( ) [stdio.h]	현재 위치 표시자가 가르키는 곳에 문자를 쓴 뒤, 위치 표시자를 다음 위치로 전진	성공 : char 실패 : EOF

3. 추가 기능 구현 방법

(1) 먼저 for문을 위한 인수 I와 현재 가장 높은 우선 순위를 기록하기 위한 인수 highest를 선언한다.

(2) 처음 들어오는 프로세스를 무조건 실행시켜야 하므로 highest를 우선순위가 가장 낮은 10으로 초기화한다.

(3) 큐에 있는 모든 프로세스가 들어올 때까지 반복해야 한다
code : for(i = 0 ; i < queue_len ; i++)

(4) 도착한 프로세스의 우선순위가 현재 진행중인 프로세스의 우선순위보다 높은지 확인
code : if(queue[i]->priority < highest)

(5) 위의 if문이 true이면 도착한 프로세스의 우선순위가 더 높으므로 해당 프로세스에 cpu할당

(6) cpu가 할당된 프로세스의 우선순위가 현재 가장 높기 때문에 highest에 우선순위 저장