고려대학교
KOREA UNIVERSITY

운영체제 2차과제 보고서

-프로세스 및 리눅스 스케줄링의 이해-

제출일	2020.06.10.	**학과**	컴퓨터학과
과목	운영체제	**학번**	2016320187
Freeday	1일 사용	**이름**	홍성윤

○ **개발환경**

Ubuntu 18.04.02 (64bit), Linux kernel 4.20.11 환경에서 진행했습니다.

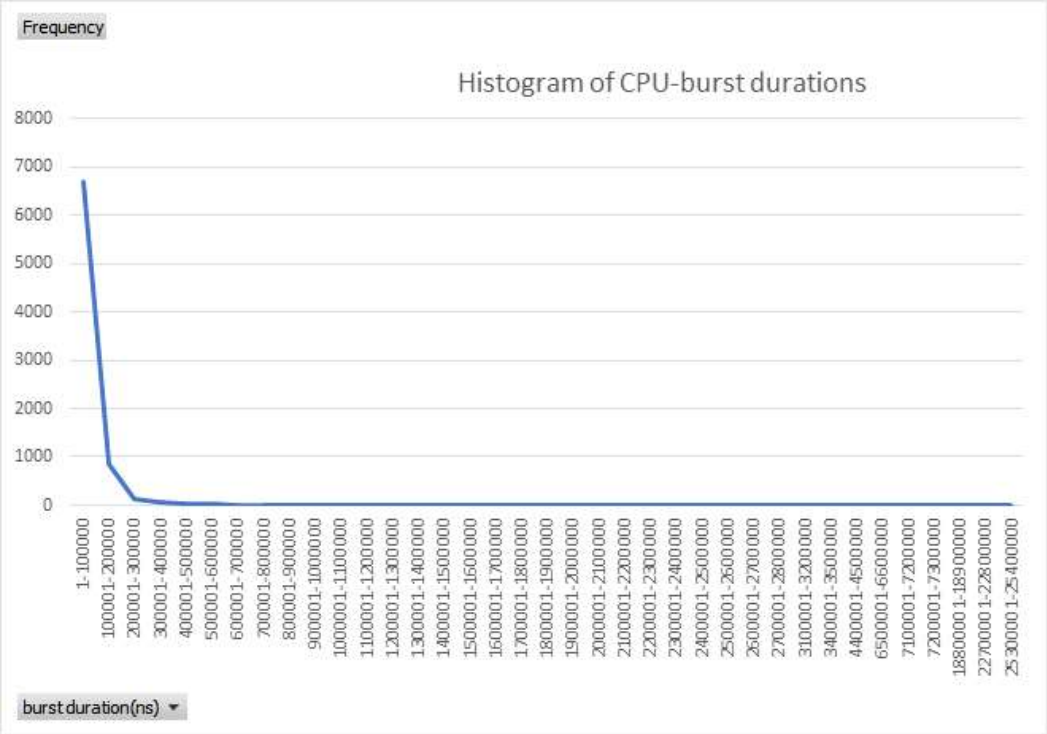
○ 과제 개요 및 프로세스와 스케줄러의 개념

프로세스는 프로그램의 abstraction 이라고 볼 수 있다. 코드로 작성되어 저장된 프로그램이 메모리에 올라가게 되면 이를 프로세스라고 한다. 메모리에 올라온 프로세스는 CPU를 할당받아 실행되게 되고, 여러 프로세스들이 서로 문맥전환 되며 CPU에서 실행된다. 프로세스는 Process control block(PCB)라는 자료구조로 표현되고 PCB는 프로세스와 관련된 여러 정보들을 포함하고 있다. 프로세스의 상태는 new, running, sleep, ready, terminated 의 상태가 있다. new 상태에서 생성된 프로세스는 ready 상태가 되고, terminated 가 될 때 까지 ready, running, sleep 상태를 반복하게 된다.

스케줄링이란 PC에서 여러 프로세스들이 수행될 때, 어떻게 프로세스에게 CPU의 사용을 분배할지 결정하는 것을 말한다. 즉, 특정 시간 t에서 어떤 프로세스에게 어떤 순서로 CPU를 할당할지 결정하는 것이다. 스케줄러가 어떻게 스케줄링을 하는지에 따라 전체성능을 좌우할 정도로 스케줄링은 매우 중요하다. CPU의 사용률과 처리량을 최대로 늘리고, 응답시간과 대기시간은 최소한으로 하는 스케줄러가 이상적인 스케줄러라고 할 수 있겠지만 현실적으로 이러한 스케줄러를 만드는 것은 불가능하다. 따라서 시스템의 용도에 따라 알맞은 스케줄러를 구성하게 된다.

CPU Burst는 CPU로 연산을 수행하는 시간이고, I/O Burst는 I/0 처리를 위해 기다리는 시간이다. 일반적으로 프로세스는 두 burst를 번갈아 가며 수행된다. 2차 과제를 통해 여러 프로세스들의 CPU burst를 측정하고 이에 대한 분석을 해보았다.

○ CPU burst에 대한 그래프 및 결과분석



다음 그래프는 구간별 CPU burst의 빈도수를 나타낸 그래프이다. 대부분의 프로세스들이 CPU burst 값이 일정 값 이하이고 그 이상의 CPU burst를 가지는 빈도수는 1 이하이거나 매우 작음을 확인할 수 있다. 실제로 서로 다른 프로세스, 시스템에도 불구하고 대체적으로 위의 그래프와 같은 경향을 지닌다. 대부분의 CPU burst가 일정 시간 이하이기 때문에 타임퀀텀을 적절히 설정한다면 대부분의 프로세스들을 퀀텀 안에 끝낼 수 있고, time sharing으로 인한 오버헤드를 줄일 수 있을 것이다.

○ 작성한 모든 소스코드에 대한 설명

sched_info_depart 함수는 프로세스가 CPU 점유를 마쳤을 때 호출되는 함수이다. delta 값은 현재 시간에서 프로세스가 CPU 점유를 시작했을 때의 시간을 뺀 값이다. sched_info_depart함수가 호출될 때의 현재시간은 프로세스가 CPU 점유를 마치는 시간이므로 delta는 CPU burst와 같은 의미라고 판단했다. 따라서 CPU burst 값으로 delta 값을 출력하도록 했다.

프로세스마다 1000회 호출시 CPU burst를 출력하기 위해서 task_struct 구조체를 사용했다. task_struct 구조체는 리눅스에서 Process control block 역할을 하는 구조체였다. task_struct 구조체 내부에 sched_info라는 구조체가 사용된 것을 확인할 수 있었는데, sched_info 구조체의 pcount 변수를 사용해 해당 프로세스가

1000n번 호출되었는지를 확인했다. task_struct 구조체의 tgid 변수를 통해 커널로그에 pid를 출력했다.

○ 과제 수행 시의 문제점과 해결 과정 또는 의문 사항

과제를 수행할 때 가장 어려웠던 부분 중 하나는 프로세스가 1000회 호출된 것을 확인하는 방법이었습니다. 그래서 계속해서 task_struct 구조체를 분석하던 중 sched_info 구조체 내용을 확인하게 되었고 '# of times we have run on this CPU'를 나타내는 pcount 변수를 이용하면 될 거라고 판단해 이 변수를 통해 프로세스 1000회 호출마다 CPU burst를 출력했습니다.