**운영체제 LAB 2 보고서**

|  |  |
| --- | --- |
| **과 목 명** | 운영체제 |
| **교 수 명** | 박문주 교수님 |
| **학 과** | 컴퓨터공학부 |
| **학번/이름** | 202101472 김형호 |
| 202101467 김유진 |
| 202101499 윤지영 |
| **제 출 일** | 2025. 5 .?. |

**팀 보고서**

* xv6 분석

추가 예정

* 설계 및 구현

uint wait\_ticks;

extern int temp\_ticks;

* Trap.c

|  |  |
| --- | --- |
| ✅ 코드 | ✅ 설명 |
|  | #include "pstat.h"을 불러오기  Proc.c 에서 선언한 temp\_ticks, pstat pst을 참조 temp\_ticks : trap에서 ticks가 상승할 때 같이 상승시켜 준 뒤 proc.c의 스케줄러에서 wait\_ticks를 상승시킬 때 사용하는 누적값 pstat 구조체의 ticks와 wait\_ticks를 변경하기 위해 pst를 참조  trap 수정내용 러닝인 프로세스일 때 ticks과 now\_ticks을 상승시키고, now\_ticks가 타임 슬라이스 이상이 될 경우 && 정책이 2번이 아닌 경우 now\_ticks와 wait\_ticks를 초기화 시키고 우선순위를 강등시킨 뒤에 양보하는 코드, 정책이 2번일 경우 ticks만 초기화 now\_ticks(한번에 사용한 ticks값을 저장하는 변수)  Now\_ticks(한번에 사용한 틱)이 타임슬라이스 이상이 될 경우 즉시 강등 관련된 정보를 출력   1. 기존 코드에서는 매 틱마다 양보를 발생시켜 스케줄러 호출, mlfq에서는 time\_slice를 보장해주어야 한다고 판단하여 제거, time\_slice를 넘는 틱을 한번에 사용할 경우 강등시키면서 양보 발생, 스스로 양보할 경우 누적 틱을 증가, 누적 틱이 10타임 슬라이스를 초과할 경우 강등시키며 양보 발생  정책이 0,2가 아니고 누적 ticks가 타임슬라이스 초과시 (치팅 발견 시) 관련된 정보를 출력하고 우선순위 강등 및 ticks 관련 데이터 초기화 2. 타임 슬라이스를 보장해주기 위해서 매 틱마다 양보하던 코드 비활성화 타임슬라이스를 초과하거나, 치팅 적발시 양보 |

* Proc.h

|  |  |
| --- | --- |
| ✅ 코드 | ✅ 설명 |
|  | sched\_policy -> 현재 스케줄링 규칙을 저장하는 변수 |

* proc.c

|  |  |
| --- | --- |
| ✅ 코드 | ✅ 설명 |
|  | Pstat.h 헤더파일 불러오기  Temp\_ticks: trap에서 상승시키고 스케줄러에서 wait\_ticks 상승을 위한 변수 생성 pst : wait\_ticks, ticks를 사용하기 위해 pstat구조체를 선언, trap과 공유  now\_ticks : 한번에 실행되는 ticks을 기록하기 위한 코드, trap와 공유  used\_process[NPROC] : 같은 우선순위 일 경우 fifo로 구동되는 프로세스를 라운드 로빈으로 만들기 위해 0과 1을 저장하는 변수, 사용된 프로세스는 1, 미사용 프로세스를 0으로 두고, 같은 우선순위일 경우 0에게 우선권을 주는 변수  sed\_process[NPROC]를 초기화  last\_slected : 마지막으로 선택된 프로스세를 저장, wait\_ticks 초기화에 사용  max\_ticks : 치팅방지 10x time\_slice를 설정하는 상수 배열  max\_wait\_ticks : 부스팅을 위한 상수 배열  if문에서 c에 저장된 cpu구조체의 sched\_policy에 따라 일부 코드 비활성화, 0일 경우 라운드 로빈으로 변경  1. 정책이 0이 아닐 경우 Trap에서 상승시킨 temp\_ticks를 모든 프로세스에 적용시키고  3. it\_ticks가 max\_wait\_ticks를 초과할 경우 우선순위를 1 상승시키고 ticks, wait\_ticks 초기화, 관련된 정보를 출력  4. 우선순위가 높은 프로세스에서부터 다음 실행 프로세스 선택 이후 goto를 통해 점프, sched\_policy가 0일 경우 라운드 로빈으로 진행 같은 우선순위일 경우 used\_process가 0인 프로세스 우선 실행, 해당 우선순위의 모든 프로세스가 used\_process==1인 경우(프로세스가 없는 경우에는 0) has\_this\_level 을 1로 업데이트, has\_this\_level  가 1인경우 해당 우선순위의 모든 프로세스의 used\_process 값을 0으로 초기화, 우선순위를 내리기 전에 현재 우선순위의 프로세스들을 실행 goto, found를 통해 이동  5. 프로세스를 실행하고, 직전에 실행된 프로스세의 now\_ticks 초기화 last\_seleted도 초기화  다른 프로세스에서 Sleep(1)을 사용하고 비치팅 프로세스에서는 사용하지 않을 경우 해당 프로세스를 제외한 모두가 상태 2(슬립)이 되어 과도하게 비치팅 프로세스가 실행되는 현상발생, 실행결과를 보여줄 수 없어, 실행결과를 보여주기 위한 코드  fork(void) 의 최하단부에 used\_process를 0으로 설정해주는 코드 추가  (새로 생성된 프로세스가 우선순위를 가지도록 한번 더 0으로 설정 ) |

* Mlfq3.c ->개인과제2

|  |  |
| --- | --- |
| ✅ 코드 | ✅ 설명 |
|  | #include "types.h"  #include "stat.h"  #include "user.h"  #include "pstat.h" 기본 적인 헤더파일들을 호출  #define NCHILD 4 프로세스를 만들 상수 선언  workload() 함수  int pid[NCHILD]; 크기 4짜리 배열 선언, pid를 담을 배열  struct pstat st;ㅋ  int i, k,ii;, 함수 내부의 for문을 위한 변수  setSchedPolicy(2)로 정책을 2로 결정(ticks 누적 w/o(치팅 허용))  fork()로 자식 프로세스 생성 pid[i]에 fork의 리턴값 저장  (자식 프로세스는 0)  for문의 i를 따라서 프로세스별 workload() 인수 변경 i가 0일 때 대략 6틱에 해당하는 400000 선정  i가 1일 때 대략 15틱에 해당하는 800000 선정  i가 2일 때 대략 30틱에 해당하는 1600000 선정  I가 3일 떄  대략 100틱에 해당하는 6000000 선정  치팅 프로세스는 for문에 10 할당 비치팅 프로세스는 for문에 200 할당 (sleep(1)을 사용해도 슬립한 동안 비치팅 프로세스만이 존재하게 되는 현상으로 인해 충분한 숫자 할당)  모든 자식 프로세스가 종료될 때 exit(); 선언   1초 기다린 후에 getpinfo 함수로 pstat 구조체 가져오기 inuse가 0이고, pid가 pid[]배열 에 있는 프로세스들을  printf 함수로 wait\_ticks, ticks, pid, inuse, priority, state 출력  wait();로 자식 프로세스가 종료되길 기다렸다가  한번 더 pstat의 데이터 출력하고  프로세스 종료 |

)