소프트웨어학부 3학년 / 20180285 / 박민준

운영체제 3차과제

개발 환경 : vmware : ubuntu 20.04 :: Linux

수행 과정

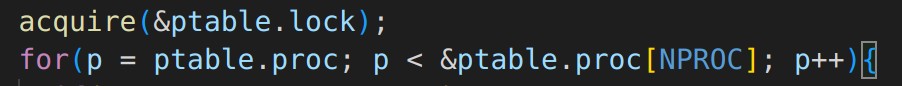
1. XV6 운영체제의 스케줄러인 void scheduler(void) 함수 분석하기

텍스트이(가) 표시된 사진

자동 생성된 설명함수 내 사용할 프로세스 선언 뒤, cpu 초기화.

스케줄러는 무한한 루프 함수.

인터럽트가 가능하도록 반복문 내 함수 호출.



Ptable에 대한 lock을 얻은 후, ptable 내의 proc를 순회

텍스트이(가) 표시된 사진

자동 생성된 설명

순회 중, p의 상태가 RUNNABLE 하다면 실행할 프로세스로 선택.

선택 후, cpu에 프로세스를 지정하고, switchuvm으로 상태를 변경한 뒤, 상태를 RUNNING으로 변경

이때, 계속해서 context switching 이 발생하며 RUNNABLE한 프로세스를 번갈아가며 실행하기 때문에 위 스케줄러는 라운드 로빈 스케줄러.

텍스트이(가) 표시된 사진

자동 생성된 설명

Context를 선정하였던 p의 context로 변경.

Kernel mode 로 변경 해준 뒤, cpu의 proc또한 초기화.

Ptable에 대한 lock을 반환.

위 과정을 반복.

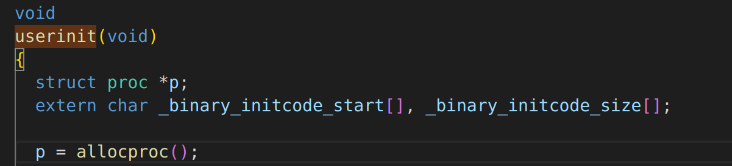
1. 각 프로세스에 “priority” 개념 추가

* proc.h 파일안에 있는 “proc” 구조체에 “int priority” 멤버 추가

텍스트, 시계이(가) 표시된 사진

자동 생성된 설명멤버변수 priority 추가.

* 프로세스가 생성될 때, priority 값은 5 이어야 함 (최초로 실행되는 프로세스부터)

최초 프로세스를 생성하는 userinit 함수에서 p = allocproc(); 를 통해 프로세스르 할당.

텍스트이(가) 표시된 사진

자동 생성된 설명allocproc 함수가 반환해주는 proc \*p 변수에 p->priority = 5; 코드를 추가하여 priority 값을 5로 지정.

* fork()를 통해서 프로세스가 생성될 때 자식 process는 부모 process 와 같은 priority 값은 가져야함

텍스트이(가) 표시된 사진

자동 생성된 설명fork 함수 에서는 자식 프로세스인 proc \*np 변수에 np의 priority는 curproc의 값과 같게 해주는 코드 추가

np ->priority = curproc->priority;

* “set\_proc\_priority” 시스템 콜 구현

텍스트이(가) 표시된 사진

자동 생성된 설명

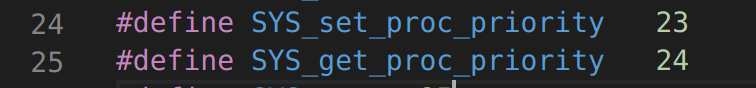
* “get\_proc\_priority” 시스템 콜 구현

텍스트이(가) 표시된 사진

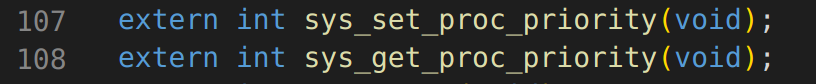
자동 생성된 설명

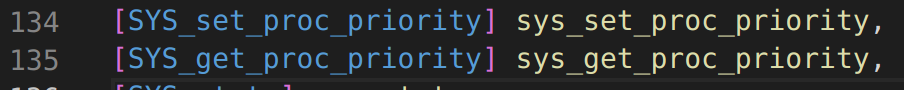
시스템 콜 등록 과정

<syscall.h>

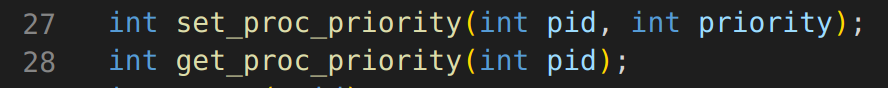


<syscall.c>





<user.h>



<usys.S>

텍스트, 표지판, 검은색이(가) 표시된 사진

자동 생성된 설명

1. void scheduler(void) 함수 수정하여 Priority Scheduler 구현



bestproc 추가

텍스트, 화면, 스크린샷이(가) 표시된 사진

자동 생성된 설명

실행할 프로세스를 priority 비교를 통해 선정.

tmp의 priority 가 더 작다면 해당 프로세스를 bestproc로 지정.

마지막으로 bestproc를 현재 실행할 p프로세스로 설정.

<테스트 시나리오>

1. 현재 프로세스의 상태를 확인할 수 있는 state시스템콜 추가.

텍스트, 화면, 스크린샷이(가) 표시된 사진

자동 생성된 설명

결과화면

텍스트이(가) 표시된 사진

자동 생성된 설명

1. Proc 구조체에 실행된 횟수를 저장하는 변수 cnt 추가.

proc.h

텍스트이(가) 표시된 사진

자동 생성된 설명 fork에서 cnt초기화

allocproc에서 cnt초기화

텍스트이(가) 표시된 사진

자동 생성된 설명

cnt는 scheduler함수 내에서 해당 프로세스로 context switching이 일어날 경우 1씩 증가함.

1. 더미 프로세스 생성

nproc 명령어 구현 – 프로세스 런타임을 늘리기 위해 다음과 같은 for문 구성.

텍스트이(가) 표시된 사진

자동 생성된 설명

nproc &; nproc &;

1. State 시스템 콜을 통해 현재 모든 프로세스의 상태를 확인

State

텍스트이(가) 표시된 사진

자동 생성된 설명

1. 실행중인 더미 프로세스 하나의 priority를 7로 변경 (set\_proc\_priority를 활용한 chpr.c 사용)

텍스트이(가) 표시된 사진

자동 생성된 설명

chpr 5 7

1. 새로운 더미 프로세스 생성 (priority는 default 5)

nproc &;

1. State 시스템 콜을 통해 priority가 변경된 프로세스의 상태 확인

State

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

---------------------proc의 cnt 변수를 통해 선점형 방식 스케줄러 확인--------------------

1. Priority가 7인 프로세스는 Starvation 상태가 됨. Priority 가 7인 프로세스의 cnt 값이 변동되지 않음을 확인.

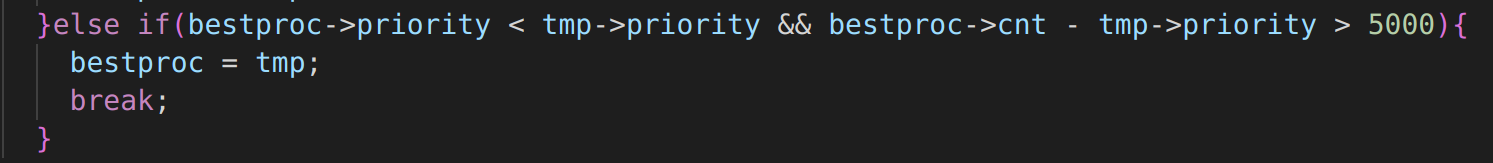
라운드로빈 스케줄러에서 실행 횟수를 저장하는 cnt 변수가 증가하지 않고 그대로인 모습이 관측됨.

텍스트이(가) 표시된 사진

자동 생성된 설명

--------------------------starvation 현상이 나타나는 모습을 직접 관측-----------------------

1. bestproc를 지정할 때 priority 우선순위가 낮고(값이 더 크고), 실행횟수(cnt)가 5000이상 차이 날 경우, priority와 관계없이 bestproc로 지정되도록 코드 작성.



1. 실행 결과.

텍스트이(가) 표시된 사진

자동 생성된 설명

두번의 state 결과 priority가 7인 프로세스가 starvation 현상이 생기지 않고 cnt가 증가함을 확인할 수 있음.

------------------------실행횟수를 기억하여 starvation 현상 자체 해결----------------------

발생한 문제점.

1. Starvation 관측 과정에서 많은 어려움을 겪음.

* Xv6의 경우 프로세스 실행 시 nproc와 같이 해주면 명령어 실행 도중 다른 명령어를 사용하기 어려움. 이에 nproc &; 같은 &인자를 더해주어 명령어 실행 중에도 다른 명령어를 사용할 수 있게 해 줌.

1. Set, get proc\_priority 구현 중 오류

* 시스템콜로 user program 에서 입력받은 pid를 전달 하는 방법을 지난 과제를 참고하여 해결. argint 함수 사용