과제 #3 : XV6에 SSU Scheduler 구현

○ 과제 목표

- xv6의 프로세스 관리 및 스케줄링 기법 이해
- xv6에서 기존 스케줄러 변경 및 명령어 추가

○ 기본 지식

- 스케줄링
  - ✓ 스케줄링은 다중 프로그래밍을 가능하게 하는 운영체제 커널의 기본 기능임
  - ✓ 기존 xv6의 스케줄링 기법은 다음 실행할 프로세스를 process table을 순회하며 RUNNABLE 상태인 프로세스를 순차적으로 선택함

○ 과제 내용

1. SSU Scheduler 구현
   - ✓ SSU Scheduler는 프로세스 실행 흐름에서 새로 생긴 프로세스가 제외되지 않도록 큰 가중치를 부여하는 스케줄러임
   - ✓ 모든 프로세스는 각자의 가중치(weight) 및 우선순위(priority) 값을 가짐
   - ✓ priority 값은 스케줄링 함수가 호출될 때마다, 다음과 같은 규칙에 따라 업데이트됨
     - new_priority = old_priority + (time_slice / weight)
     - 해당 과제에서는 time_slice 값을 10,000,000 ns로 사용
   - ✓ SSU Scheduler를 위한 스케쥴링 함수는 다음 실행될 프로세스로 RUNNABLE 상태인 프로세스 중 가장 낮은 priority 값을 가진 프로세스를 선택함
   - ✓ weight 값은 프로세스 생성 순서에 따라 1부터 차례대로 증가시키며 부여함
   - ✓ 프로세스 생성 또는 wake up 시 priority 값을 0부터 부여하게 되면, 해당 프로세스가 독점 실행될 수 있으므로 관리하고 있는 프로세스의 priority 값 중 가장 작은 값을 부여함
     - xv6에서 프로세스 wake up은 프로세스 상태가"SLEEPING"에서 "RUNNABLE"로 전이되는 것을 의미함
     - 가장 작은 priority 값은 struct ptable의 멤버로 관리하며, 스케줄링 함수가 호출될 때마다 new_priority과 함께 갱신됨
2. 구현한 스케쥴링 함수의 동작 과정을 확인하는 sdebug 명령어 및 이를 위한 weightset() 시스템 콜 구현
   - ✓ 쉘에 "sdebug" 입력 시 스케줄링 함수의 동작 과정을 확인할 수 있도록 프로세스 ID, 프로세스의 가중치, 프로세스 생성 후부터 해당 명령어에 의해 프로세스 정보가 출력되기까지 소요된 시간을 출력하는 기능 구현
   - ✓ 해당 명령어 구현을 위해 필요한 매크로 선언은 다음과 같음
     - "PNUM"
       - ☞ 구현한 스케줄링 함수를 확인하기 위해 fork()로 생성할 프로세스의 개수
       - ☞ 기본값 : 5
     - "PRINT_CYCLE"
       - ☞ fork()로 생성된 프로세스 중 실행 중인 프로세스 정보를 출력하는 주기
       - ☞ 프로세스 별로 한 번만 출력됨
       - ☞ 기본값 : 100000000
     - "TOTAL_COUNTER"
       - ☞ fork()로 생성된 프로세스가 생성된 이후부터 소모할 수 있는 시간
       - ☞ "TOTAL_COUNTER" 값을 전부 소모하면 프로세스는 종료됨
       - ☞ 기본값 : 500000000
   - ✓ "PNUM"값을 변경해도 정상적으로 동작하여야 함
   - ✓ 해당 명령어는 생성한 프로세스가 모두 종료된 후 종료되도록 구현
     - 다른 xv6 명령어 구현 방식 참고
   - ✓ weightset() 시스템 콜은 매개변수로 입력받은 값을 sdebug 명령어에 의해 생성되는 프로세스의 weight 값으로 부여하도록 구현

☞ 매개변수로 weight 값을 0으로 입력받을 시 에러처리

- sdebug 명령어에 의해 생성되는 프로세스의 weight 값은 시스템에서 생성한 프로세스와 별도로 생성된 순서에 따라 1부터 증가시키며 부여함

☞ 예시 1 참고

✓ xv6에 명령어를 추가하기 위한 일부 과정은 예시 2, 3과 같음

(예시 1). fork 하는 프로세스가 5개 일 때, sdebug 명령어 실행 예시

```
$ sdebug
start sdebug command
PID: 8, WEIGHT: 5, TIMES : 600 ms
PID: 7, WEIGHT: 4, TIMES : 750 ms
PID: 6, WEIGHT: 3, TIMES : 1040 ms
PID: 5, WEIGHT: 2, TIMES : 1580 ms
PID: 4, WEIGHT: 1, TIMES : 2980 ms
PID: 8 terminated
PID: 7 terminated
PID: 6 terminated
PID: 5 terminated
PID: 4 terminated
end of sdebug command
$ sdebug
start sdebug command
PID: 14, WEIGHT: 5, TIMES : 610 ms
PID: 13, WEIGHT: 4, TIMES : 790 ms
PID: 12, WEIGHT: 3, TIMES : 1030 ms
PID: 11, WEIGHT: 2, TIMES : 1570 ms
PID: 10, WEIGHT: 1, TIMES : 3010 ms
PID: 14 terminated
PID: 13 terminated
PID: 12 terminated
PID: 11 terminated
PID: 10 terminated
end of sdebug command
$
```

(예시 2). sdebug.c 구현

```
#include “types.h“
#include “stat.h“
#include “user.h“

#define PNUM 5
#define PRINT_CYCLE 100000000
#define TOTAL_COUNTER 500000000

void sdebug_func(void)
{
  // 구현
  return ;
}

int main(void)
{
  sdebug_func();
  exit();
}
```

(예시 3). Makefile 수정

```
(생략)
UPROGS=\
	_cat\
	_echo\
	_forktest\
	_grep\
	_init\
	_kill\
	_ln\
	_ls\
	_mkdir\
	_rm\
	_sh\
	_stressfs\
	_sdebug\
	_usertests\
	_wc\
	_zombie\

(생략)

EXTRA=\
	mkfs.c ulib.c user.h cat.c echo.c forktest.c grep.c kill.c\
	ln.c ls.c mkdir.c rm.c sdebug.c stressfs.c usertests.c wc.c zombie.c\
```

```
    printf.c umalloc.c \
    README dot-bochsrc *.pl toc.* runoff runoff1 runoff.list\
    .gdbinit.tmpl gdbutil\
```

3. 다음에 실행될 프로세스 선정 과정 디버깅 기능 구현

- ✓ 예시 4와 같이 xv6 빌드 시 "debug=1"매개변수 전달을 통해, 스케줄링 함수에서 다음 실행될 프로세스를 선택할 때마다 PID, 프로세스 이름, 프로세스 가중치, 프로세스 우선순위 값을 출력하도록 구현
- ✓ 다음 프로세스를 선택할 수 없는 경우에는 출력하지 않음
- ✓ 이에 대한 빌드 및 실행 예시는 예시 4와 같음

(예시 4). 3번 기능 실행 예시

```
$ make debug=1 qemu-nox

(build)

Booting from Hard Disk..xv6...
cpu0: starting 0
PID: 1, NAME: initcode, WEIGHT: 1, PRIORITY: 3
PID: 1, NAME: initcode, WEIGHT: 1, PRIORITY: 3
(생략)
PID: 1, NAME: init, WEIGHT: 1, PRIORITY: 10000003
PID: 1, NAME: init, WEIGHT: 1, PRIORITY: 20000003
PID: 1, NAME: init, WEIGHT: 1, PRIORITY: 20000003
(생략)
init: starting sh
PID: 2, NAME: init, WEIGHT: 2, PRIORITY: 20000003
PID: 2, NAME: init, WEIGHT: 2, PRIORITY: 25000003
PID: 2, NAME: init, WEIGHT: 2, PRIORITY: 25000003
$ sdebug
PID: 2, NAME: sh, WEIGHT: 2, PRIORITY: 25000003
PID: 2, NAME: sh, WEIGHT: 2, PRIORITY: 30000003
PID: 3, NAME: sh, WEIGHT: 3, PRIORITY: 25000003
PID: 3, NAME: sh, WEIGHT: 3, PRIORITY: 25000003
(생략)
start sdebug command
start sdebug command
PID: 3, NAME: sdebug, WEIGHT: 3, PRIORITY: 28333336
PID: 4, NAME: sdebug, WEIGHT: 4, PRIORITY: 25000003
PID: 5, NAME: sdebug, WEIGHT: 5, PRIORITY: 28333336
PID: 6, NAME: sdebug, WEIGHT: 6, PRIORITY: 28333336
PID: 3, NAME: sdebug, WEIGHT: 3, PRIORITY: 31666669
PID: 7, NAME: sdebug, WEIGHT: 7, PRIORITY: 28333336
PID: 7, NAME: sdebug, WEIGHT: 4, PRIORITY: 30833336
(생략)
PID: 8, WEIGHT: 5, TIMES: 700 ms
(생략)
PID: 7, WEIGHT: 4, TIMES: 900 ms
```

```
(생략)
PID: 6, WEIGHT: 3, TIMES: 1150 ms
(생략)
PID: 5, WEIGHT: 2, TIMES : 1730 ms
(생략)
PID: 4, WEIGHT: 1, TIMES : 3430 ms
(생략)
PID: 8 terminated
(생략)
PID: 7 terminated
(생략)
PID: 6 terminated
(생략)
PID: 5 terminated
(생략)
PID: 4 terminated
PID: 3, NAME: sdebug, WEIGHT: 3, PRIORITY: 1256666669
PID: 3, NAME: sdebug, WEIGHT: 3, PRIORITY: 1260000002
end of sdebug command
PID: 2, NAME: sh, WEIGHT: 2, PRIORITY: 1260000002
$
```

○ 기타 참고사항

- 본 과제는 SSU Scheduler를 위한 스케쥴링 함수 구현이 목표이므로, CPU 코어 개수를 1개로 제한
  - ✓ Makefile 수정 필요
- init process가 처음 생기는 시점에서 최소 priority 값을 3으로 지정
  - ✓ 시스템 시작 시 총 세 개의 유저 프로세스(“initcode”, “init”, “sh”)가 생성되기 때문에 최소 priority 값을 3으로 지정

○ 과제 제출 마감

- 2022년 10월 19일 (수) 23시 59분 59초까지 구글 클래스룸으로 제출
- 보고서 (hwp, doc, docx 등으로 작성) 기존 스케쥬링 관련 코드 분석과 이를 어떻게 수정했다는 내용이 반드시 포함되어야 함
- xv6에서 변경한 소스코드 (실행파일 제출하지 말 것)
- 1일 지연 제출마다 30% 감점. 4일 지연 제출 시 0점 처리 (이하 모든 설계 과제 동일하게 적용)

○ 필수 구현

- 1, 2

○ 배점 기준

- 1 : 50점
- 2 : 40점
- 3 : 10점