In the name of god

OS project
phase_2 report
Dadfar Mo’meni
965222006

---

attention:
2.I tried to keep up with instructions in “readme.md” file of following project on “github”:
https://github.com/prakhar987/xv6-system-calls.git

STEP 0:

1. same as phase_1: initialize xv6 project git repository and install qemu.

2. in order to add the system call to XV6 we only have to make changes to these 5 following files:

    1. syscall.h
    2. syscall.c
    3. sysproc.c
    4. proc.h
    5.proc.c
    6.defs.c
    7.trap.c
    8. usys.S
    9. user.h
    10. Makefile

and create a c file to test the system call in our case :

scheduling_test_file.c

STEP 1:

1. give an index to the system call in syscall.h

    #define SYS_scheduling 24

2. Add pointer to the system call in syscall.c

    [SYS_scheduling] sys_scheduling,

When the system call with number 24 is called by a userprogram, the function pointer sys_scheduling which has the index SYS_scheduling or 24 will call the system call function.

STEP2:

1. to implement the system call function, first we have to add function prototype in syscall.c:

extern int sys_scheduling(void);

2. to implement the system call function we use the wait function in proc.c and make some changes to it.We use sysproc.c file to call the scheduling function implementaion present in proc.c.

sys_scheduling function in sysproc.c although does one task for us: it passes the parameters to the original funcion of proc.c

3. in order to calculate the sleeping time and runnig time we should add some lines to proc.h and trap.c

4.how start time , sleep time , running time and I/O time are calculated:

*starting time is recoreded in allocproc() function of proc.c.(When process is born)

*etime is recorded in exit() function (i.e when child exists, ticks are recorded) of proc.c.

*rtime is updated in trap() function of trap.c .(IF STATE IS RUNNING , THEN UPDATE rtime)

*iotime is updated in trap() function of trap.c.(IF STATE IS SLEEPING , THEN UPDATE wtime)

STEP 3:

1. In order for a user program to call the system call, an interface needs to be added. Therefore, we need to edit the usys.S file.

SYSCALL(scheduling)

2. Next, the user.h file needs to be edited.

```
int scheduling(int *,int *);
```

STEP 4:

1. in order to test the functionality of the system call, we need to add a user program which calls this system call.
scheduling_test_file.c

STEP 5:

1. to test the system call we need to run these commands in terminal where all xv6 files are located.

$ make clean
$make
$make qemu

2. to run our test program we simply run this command in qemu terminal:
./scheduling_test_file