CMPE 322

Modifying the Process Scheduling in Minix

Gönül AYCI
2016800003
December 27, 2016

1 Modifying and Compiling the Minix Kernel

In this part of the project, first we need to backup the original source files into a different directory. I used recursive copy operation in Minix with `cp -r src src_orig`. Then I modify src source file and save the original file which is `src_orig`. In this project, I continue with first project src file.

2 Implementing the Lottery scheduling algorithm with kernel torpil

2.1 The system calls

2.1.1 table.c

Firstly, we need to add an entry to the PM server system call table that maps a call number. `table.c` file in path `/usr/src/minix/servers/pm/`. Then, I added

- `CALL(SETTICKETS) = do_settickets`
- `CALL(SETTORPIL) = do_settorpil`

at the end of the file.

2.1.2 CALLNR.H

Secondly, we need to add a call number definition for the table entry added in the previous step. I did it in `callnr.h` file which is in `/usr/src/minix/include/minix`. I added

- `#define SETTICKETS (PM_BASE + 49)`
- `#define SETTORPIL (PM_BASE + 50)`

When I added these lines at the end of the definition part, I had a problem because of highest number from base plus one condition. I organized it that I assigned a call number 49 and 50.

2.1.3 PROTO.H

Thirdly, I need to define the system call's function prototype. I used `proto.h` file which is in `/usr/src/minix/servers/pm/`. I added

- `int do_settickets(void)`
- `int do_settorpil(void)`

lines of code to `misc.c` part of this header file.

2.1.4 DO_SETTICKETS.C AND DO_SETTORPIL.C

I created c files `do_settickets.c` and `do_settorpil` which are doing `taskcall` `SCHEDULING_SETTICKETS` and `SCHEDULING_SETTORPIL`.

2.1.5 MAKEFILE

I need to add the name of the C file to the list of SRCS to compile with the PM server so, I read `Makefile` with `vi`. `Makefile` is in `/usr/src/minix/servers/pm/`. I added my `do_settickets.c do_settorpil.c` file at the end of the SRCS part.

2.1.6 SYS

I implement two system calls in `PM` that each of them have two parameters. These are:

- `int settickets(int val, int pid)`
- `int settorpil(int val, int pid)`

Additionally, I modify `Makefile.inc` that adding our system calls which are `settickets.c settorpil.c`

2.1.7 WRITE A LIBRARY

I wrote a user library function for the system call. I followed these steps:

First, I edited `unistd.h` file which is in `/usr/src/include/`. I added a line which is `int settickets(int value, int pid);` and `int settorpil(int value, int pid);` before `__END_DECLS` and at the end of this header file.

2.2 IN THE USER SPACE AND KERNEL SPACE SCHEDULER

2.3 SCHED/MAIN.C

- `SCHEDULING_SETTICKETS` –> get message
- `SCHEDULING_SETTORPIL` –> get message

2.3.1 SCHEDULE.C

Define queues

- `define WINNER 12`
- `define LOSER 13`
- `define TORPIL 14`

In additionally, I add a function which is used for detecting that is the user `int is_user(int prior)`
I define `int sched_settickets(message *m_ptr)` and `int sched_settorpil(message *m_ptr)` function in the scheduling c file. In addition to this, I write a `do_lottery()` function which is useful for schedule processes.

- To call `do_lottery()` into `do_noquantum` function.
- `start_scheduling`
- `do_noquantum`
- to convert passive state `balance_queues`

2.3.2 SCHEDPROC.H

Add

- `unsigned max_tickets`
- `unsigned torpil`
- `unsigned num_tickets`
- `unsigned min_tickets`

2.3.3 PROTO.H

I add to call

- `int sched_settickets(message *m_ptr)`
- `int sched_settorpil(message *m_ptr)`

2.3.4 KERNEL

In `proc.c` file, I define queues which are described in

- `define WINNER 12`
- `define LOSER 13`
- `define TORPIL 14`

I also modify `pick_proc` in terms of `TORPIL` queue.

3 TESTING THE ALGORITHM

- `user add -m -g users gnl`
- `su gnl`
- `chmod +x ioBoundTest.sh`
- `./cpuBoundTest 1 100`
- `./ioBoundTest.sh`
- `cat result*`

In the testing step, `./cpuBoundTest 1 100` provides a test case for only one process which is `TORPIL` and has 100 tickets.

CPU BOUND TEST results:

- Time spent for 1: 5
- Time spent for 2: 5
- Time spent for 3: 9
- Time spent for 4: 14
- Time spent for 5: 17

CPU BOUND TICKET TEST results:
When I `./cpuBoundTicketTest 6 100`, it means that processor 6 gets 100 tickets.

- Time spent for 1: 3
- Time spent for 2: 4
- Time spent for 3: 4
- Time spent for 4: 4
- Time spent for 5: 11

CPU BOUND TORPIL TEST results:
When I `./cpuBoundTorpilTest 6 1`, I expect to get processor 6 fastly complete its tickets.

- Time spent for 1: 8
- Time spent for 2: 8
- Time spent for 3: 4
- Time spent for 4: 16
- Time spent for 5: 5

IO BOUND TEST results:

- Time spent for 1: 11
- Time spent for 2: 13
- Time spent for 3: 13
- Time spent for 4: 13
- Time spent for 5: 12

IO BOUND TICKET TEST results:
When write io bound ticket test for five processors which last one has a torpil.

- Time spent for 1: 8
- Time spent for 2: 7
- Time spent for 3: 8
- Time spent for 4: 8
- Time spent for 5: 6

4 FINAL COMMENTS

I use `make hdboot` and `reboot` many times. But, I did not add it many times (I ignore these steps).

I get my modification from `git diff > patch`

I have a trouble with Minix. My system has panicked at my Testing step. I lost so much time to recover my system.