CMPE 322

Exploring Minix and Implementing a Simple System Call

Gönül AYCI
2016800003
November 17, 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`.

1.1 MODIFICATION IN MAIN.C

In the first part of the first question requires that to find the part of banner information at the startup and edit it. So, it points to `main.c` file which is in `/usr/src/minix/kernel/`. Firstly, In this step, I found the *announce* section in `main.c`, then I added a line which includes **This kernel is modified by AYCI, GONUL, 2016800003 for this course Cmpe322**. I used `vi` for reading and editing this file. After `vi main.c` step, I used `esc+A` for writing and `esc+X` for deleting letter.

1.2 MODIFICATION IN CONFIG.H

For this part, I edited `config.h` file which is in `/usr/src/minix/include/minix/`. In this file, we have definition part at the top of this file. I edited `OS_RELEASE` line to `OS_RELEASE "4.0"` because fifth definiton line has `OS_VERSION OS_NAME " " OS_RELEASE " ("OS_CONFIG")"` information. Then, I used a similar technique of the previous part to save this edition.

1.3 MODIFICATION IN MAIN.C

In the last question of the first part, we need to locate the file that is responsible for reading the boot image and process queue at the boot. This is in `main.c` of `kmain` part. I found one of the comment line which includes *Set up proc table entries for processes in boot image.* line and I modify this part. Firstly, I added `printf("%d", NR_BOOT_PROCS)` because I need to print the number of each process involved in the boot process. Then, I added a line `printf("initializing %s... \n", ip→proc_name)` because I also need to print the total number of processes involved in the boot process.

2 IMPLEMENTING A SIMPLE MINIX SYSTEM CALL

In this problem, we need to implement a system call as a part of the process manager (PM) server with the name **printinteger**.

2.1 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(PRINTINTEGER) = do_integer` at the end of the file.

2.2 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 PRINTINTEGER (PM_BASE + 48)`. When I added this line at the end of the definition part, I had a problem because of highest number from base plus one condition which refers to 49. I organized it that I assigned a call number 48 instead of 49.

2.3 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_integer(void);` line of code to `misc.c` part of this header file.

2.4 DO_INTEGER.C

I created a file which is `do_integer.c`. In this file, we have `printf("System call PRINTINTEGER called with %d(AYCI, GONUL, %d)\n", i, 2016800003);` which is requested from part 1 of second question.

2.5 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_integer.c file at the end of the SRCS part.
Then, I compiled my system call. I went to the directory of /usr/src/releasetools/, then wrote make hdboot and reboot the system.

2.6 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 pcall(int value) before __END_DECLS and at the end of this header file.
pcall.c file, returns return _syscall(PM_PROC_NR, PRINTINTEGER, &m) which is requested from part 2 of this question. I added include <lib.h> and include <unistd.h> to pcall.c. In addition to this, pcall has an integer value.

2.7 TEST

I added a test file which name is test.c.

- Go to cd /usr/src/
- write clang test.c
- write ./a.out 5

Then I see System call PRINTINTEGER called with 5(AYCI, GONUL, 2016800003)

Finally, I created a patch file which name is my_patch. I used diff -r src_orig src > my_patch on path /usr/. In this path, src_orig is my original(clean) source directory, and src is my modified one.

3 FINAL COMMENTS

When I downloaded and run Minix before this project, I studied on Minix for training. So, I changed University part in main.c file.

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