

CS342301: Operating System

MP1: System Call

Deadline: 2024/10/14 23:59

I. Goal

1. Understand how to work in a Linux environment.
2. Understand how system calls are implemented by OS.
3. Understand the difference between user mode and kernel mode.

II. Assignment

1. Trace code

- Working items:

- (a). Trace the **SC_Halt** system call to understand the implementation of a system call.
(Sample code: halt.c)

```
machine/mipssim.cc  
Machine::Run()  
Machine::OneInstruction()
```

```
machine/machine.cc  
Machine::RaiseException()
```

```
userprog/exception.cc  
ExceptionHandler()
```

```
userprog/ksyscall.h  
SysHalt()
```

```
machine/interrupt.cc  
Interrupt::Halt()
```

- (b). Trace the **SC_Create** system call to understand the basic operations and data structure in a file system. (Sample code: createFile.c)

```
userprog/exception.cc  
ExceptionHandler()
```

```
userprog/ksyscall.h  
SysCreate()
```

```
filesystem/filesys.h  
FileSystem::Create()
```

- (c). Trace the **SC_PrintInt** system call to understand how NachOS implements asynchronous I/O using Callback functions and register schedule events. (Sample code: add.c)

\$../build.linux/nachos -d + -e add

userprog/exception.cc	ExceptionHandler()
userprog/ksyscall.h	SysPrintInt()
userprog/synchconsole.cc	SynchConsoleOutput::PutInt() SynchConsoleOutput::PutChar()
machine/console.cc	ConsoleOutput::PutChar()
machine/interrupt.cc	Interrupt::Schedule()
machine/mipsim.cc	Machine::Run()
machine/interrupt.cc	Machine::OneTick()
machine/interrupt.cc	Interrupt::CheckIfDue()
machine/console.cc	ConsoleOutput::Callback()
userprog/synchconsole.cc	SynchConsoleOutput::Callback()

Note: The actual execution flow in NachOS **may differ from the flow described above**, as NachOS is a simulated operating system. Explanation to the pseudo function call above is sufficient enough.

- (d). Trace the Makefile in code/test/Makefile to understand how test files are compiled.

- Requirements:

Include the following answers in your writing report:

- (a). Explain the purposes and details of each function call listed in the code path above.
- (b). Explain how the arguments of system calls are passed from user program to kernel in each of the above use cases.

2. Implement four I/O system calls in NachOS

- Working items:

- (a). `OpenFileId Open(char *name);`

- Open a file with the name, and return its corresponding `OpenFileId`.

- Return -1 if it fails to open the file.**

- (b). `int Write(char *buffer, int size, OpenFileId id);`

- Write “size” characters from the buffer into the file, and return the number of characters actually written to the file.

- Return -1, if it fails to write the file.**

- (c). `int Read(char *buffer, int size, OpenFileId id);`

- Read “size” characters from the file to the buffer, and return the number of characters actually read from the file.

- Return -1, if it fails to read the file.**

- (d). `int Close(OpenFileId id);`

- Close the file with id.

- Return 1 if successfully close the file. Otherwise, return -1.**

- Need to delete the `OpenFile` after you close the file**

- Requirements:

- (a). **Must maintain `OpenFileTable` and use the table entry number of `OpenFileTable` as the `OpenFileId`**

- (b). Must handle invalid file open requests, including the non-existent file, exceeding the opened file limit (at most 20 files), **duplicate file opening**, etc.

- (c). Must handle invalid file read, write, close requests, including invalid id, etc.

- (d). DO NOT modify the declaration of `OpenFileTable`, including the size.

- (e). DO NOT use any IO functions from standard libraries (e.g. **`printf()`, `cout`, `fopen()`, `fwrite()`, `write()`**, etc.).

- (f). DO NOT change any code under “machine/” folder

- (g). DO NOT modify the content of `OpenFileTable` outside “filesystem/” folder

- Hint & Reminder:

- (a). We use the stub file system for this homework, so DO NOT change or remove **the flag `-DFILESYS_STUB` in the Makefile under `build.linux/`**.

- Verification:

First use the command “`../build.linux/nachos -e fileIO_test1` ” to write a file.

Then use the command “`../build.linux/nachos -e fileIO_test2`” to read the file

```
[test@lsalab test]$ ../build.linux/nachos -e fileIO_test2
fileIO_test2
Passed! ^ ^
Machine halting!

This is halt
Ticks: total 777, idle 0, system 110, user 667
Disk I/O: reads 0, writes 0
Console I/O: reads 0, writes 0
Paging: faults 0
Network I/O: packets received 0, sent 0
```

3. Report

- Working items:
 - (a). Cover page, including team member list, team member contributions
 - (b). Explain how system calls work in NachOS as requested in Part II-1.
 - (c). Explain your implementation as requested in Part II-2.
 - (d). What difficulties did you encounter when implementing this assignment?
 - (e). Any feedback you would like to let us know.

II. Instructions

Below are the basic instructions. More information can be found in the NachOS tutorial slides or [github](#).

1. Set VPN
 - <https://hackmd.io/@0lvPWgbSSnuawqRf7BFC6Q/Hkyr5abpC>
 - **Notice:** each profile only allows **1 active connection**.
2. Login server (Alternatively, you can choose to **run Nachos locally using Docker**. See [tutorial](#).)
 - 192.168.5.2
 - Username: os24team + your teamID (e.g. os24team01)
 - Password: You are required to reset the password once you login
 - **Note:** Remember to back up your code in case of a power failure.
3. Copy project folder

You can choose **either way** to copy the source project

 - a. `cp -r /home/os2024/share/NachOS-4.0_MP1 .`
 - b. `git clone https://github.com/NTHU-LSALAB/NachOS-4.0_MP1_src.git`
4. Install NachOS
 - `cd NachOS-4.0_MP1/code/build.linux` **or** `cd NachOS-4.0_MP1_src/code/build.linux`
 - `make clean`
 - `make`
5. Compile/Rebuild NachOS
 - `cd NachOS-4.0_MP1/code/build.linux` **or** `cd NachOS-4.0_MP1_src/code/build.linux`
 - `make clean`
 - `make`

6. Test NachOS

- `cd NachOS-4.0_MP1/code/test` **or** `cd NachOS-4.0_MP1_src/code/test`
- `make clean`
- `make halt`
- `../build.linux/nachos -e halt`

IV. Grading

1. Implementation correctness – 50%
 - Pass the public and hidden test cases.
 - You **DO NOT** need to upload NachOS code to eeclass, and just put your code to the folder named “**NachOS-4.0_MP1**” in your home directory.
 - **Your working directory will be copied for validation after the deadline.**
2. Report – 30%
 - Name the report “**MP1_report_[GroupNumber].pdf**”, and upload it to eeclass.
3. Demo– 20%
 - We will ask several questions about your codes.
 - **Demo will take place on our server, so you are responsible to make sure your code works on our server.**

***Late submissions will not be accepted.** Refer to the course syllabus for detailed homework rules and policies.