

# CS342301: Operating System

## MP1: System Call

Deadline: 2019/10/20 23:59

### I. Goal

1. Understand how to work in 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)

1.	machine/mipssim.cc <b>Machine::Run()</b> <b>Machine::OneInstruction()</b>	✓
2.	machine/machine.cc <b>Machine::RaiseException()</b>	✓
3.	userprog/exception.cc <b>ExceptionHandler()</b>	✓
4.	userprog/ksyscall.h <b>SysHalt()</b>	✓
5.	machine/interrupt.cc <b>Interrupt::Halt()</b>	✓

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

1.	userprog/exception.cc <b>ExceptionHandler()</b>	✓
2.	userprog/ksyscall.h <b>SysCreate()</b>	✓
3.	filesys/filesys.h <b>FileSystem::Create()</b>	✓

- (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)

1.	userprog/exception.cc <b>ExceptionHandler()</b>	✓
2.	userprog/ksyscall.h <b>SysPrintInt()</b>	✓
3.	userprog/synchconsole.cc <b>SynchConsoleOutput::PutInt()</b> <b>SynchConsoleOutput::PutChar()</b>	✓
4.	machine/console.cc <b>ConsoleOutput::PutChar()</b>	✓
5.	machine/interrupt.cc <b>Interrupt::Schedule()</b>	✓
6.	machine/mipsim.cc <b>Machine::Run()</b>	✓
7.	machine/interrupt.cc <b>Machine::OneTick()</b>	✓
8.	machine/interrupt.cc <b>Interrupt::CheckIfDue()</b>	✓
9.	machine/console.cc <b>ConsoleOutput::CallBack()</b>	✓
10.	userprog/synchconsole.cc <b>SynchConsoleOutput::CallBack()</b>	✓

- 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 returns its corresponding `OpenFileId`.

**Return -1 if fail 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 fail 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 fail to read the file.**

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

Close the file with id.

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

- Requirements:

(a). **Must use the table entry number of `fileDescriptorTable` as the `FileId`.**

(b). **Must handle invalid file open requests, including the non-existent file, exceeding opened file limit (i.e., 20), etc.**

(c). All valid file open requests must be accepted if the opened file limit (i.e., 20) is not reached.

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

(e). **DO NOT** change any code under “machine/” folder

(f). **DO NOT** modify the content of `fileDescriptorTable` outside “filesystem/” folder

- Hint & Reminder:

(a). You can use the file operations defined in **lib/sysdep.cc**

(b). 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.

1. Login server
  - 140.114.78.227 port:22
  - Username: 2019osteam + your teamID (e.g. 2019osteam01)
  - Passwd: You are required to reset the password once you login
2. Install NachOS
  - `cp -r /home/os2019/share/NachOS-4.0_MP1 .`
  - `cd NachOS-4.0_MP1/code/build.linux`
  - `make clean`
  - `make`
3. Compile/Rebuild NachOS
  - `cd NachOS-4.0_MP1/code/build.linux`
  - `make clean`
  - `make`
4. Test NachOS
  - `cd NachOS-4.0_MP1/code/test`
  - `make clean`
  - `make halt`
  - `../build.linux/nachos -e halt`

#### IV. Grading

1. Implementation correctness – 50%
  - Pass all the test cases.
  - You **DO NOT** need to upload NachOS code to iLMS, just create and put your code to the folder named “**2019osteam[GroupNumber]/MP1**” in your home directory .
  - **Your working folder will be locked after deadline.**
2. Report – 30%
  - Name the report “**MP1\_report\_[GroupNumber].pdf**”, and upload it to iLMS.
3. Demo– 20%
  - We will be asked 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.**