



NachOS-MP1

Lecturer: Jerry Chou
TA: Angus Lin, Jerry Tung
National Tsing Hua University

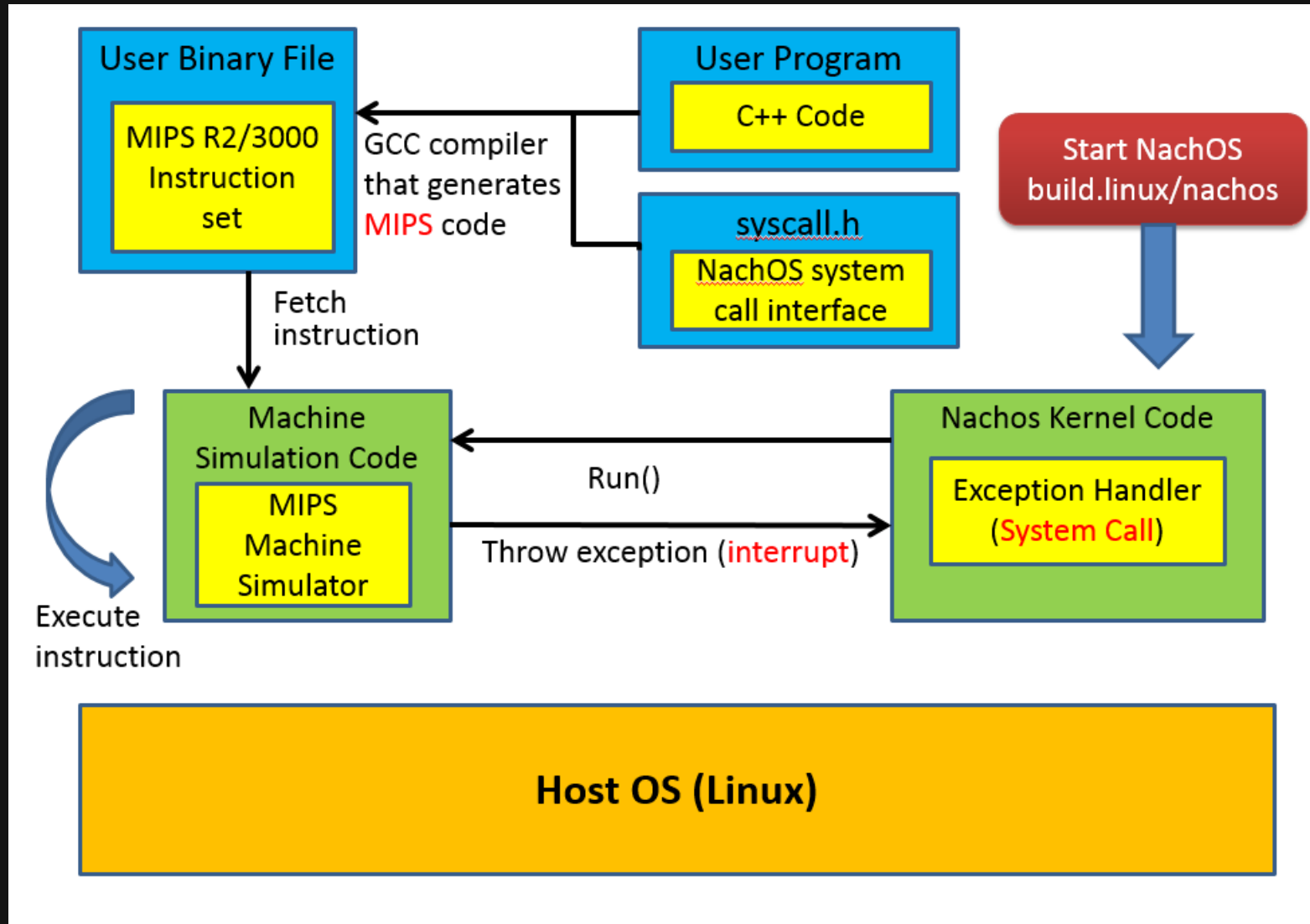
Outline

- Introduction
- Installation
- MP - System call
- Assignment
- Grading
- Hint
- FAQ
- Reference

Introduction

- Goal of this MP
 - understand how to work on Linux machine
 - understand how system call are done by OS
 - understand the difference of user space and kernel space memory
- NachOS
 - a process runs on top of another OS
 - a kernel (OS) and MIPS code machine simulator

Introduction



Nachos Directory Structure

- lib/
 - Utilities used by the rest of the Nachos code
- machine/
 - The **machine simulation**.
- threads/
 - Nachos is a multi-threaded program. Thread support is found here. This directory also contains the **main() routine of the nachos program, in main.cc.**

Nachos Directory Structure code/

- test/
 - **User test programs** to run on the simulated machine. As indicated earlier, these are separate from the source for the Nachos operating system and workstation simulation. This directory contains its own Makefile. The test programs are very simple and are written in C rather than C++.
- userprog/
 - **Nachos operating system code** to support the creation of address spaces, loading of user (test) programs, and execution of test programs on the simulated machine. The exception handling code is here, in exception.cc.

Nachos Directory Structure code/

- network/
 - Nachos operating system support for networking, which implements a simple "post office" facility. Several independent simulated Nachos machines can talk to each other through a simulated network. Unix sockets are used to **simulate network connections among the machines**.
- fileys/
 - Two different file system implementations are here. **The "real" file system** uses the simulated workstation's simulated disk to hold files. **A "stub" file system** translates Nachos file system calls into UNIX file system calls. makefile.

Installation of Nachos

- IP address: 140.114.78.227 port:22 (ssh)
 - Account: 2017osteam + **your teamID** (e.g. 2017osteam01)
 - Passwd: You will be ask to set up your password once you login
 - contact TA if you have problem logging in
- Installation (under your **home directory**)
 - `>cp -r /home/os2017/shared/NachOS-4.0_MP1 .`
 - `>cd NachOS-4.0_MP1/code/build.linux`
 - `>make clean`
 - `>make`

Installation of Nachos

- Rebuild
 - You should **rebuild NachOS** every time after you modify anything in NachOS, otherwise you won't change the execution results.
 - `>cd NachOS-4.0_MP1/code/build.linux`
 - `>make clean`
 - `>make`

Installation of Nachos

- Test your nachos
 - `>cd NachOS-4.0_MP1/code/test`
 - `>make clean`
 - `>make halt`
 - `>../build.linux/nachos -e halt`

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

This is halt
Ticks: total 52, idle 0, system 40, user 12
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
```

Installation of Nachos

- Test nachos with test cases
 - `>make` (this will generate the binary of all test cases)
 - `>make consoleIO_test1` (generate test case)
 - `>../build.linux/nachos -e consoleIO_test1` (run nachos with test case)

MP – System call

- Part I: console I/O system call
- consoleIO_test1.c

```
1 #include "syscall.h"
2
3 int
4 main()
5 {
6     int n;
7     for (n=9;n>5;n--) {
8         PrintInt(n);
9     }
10    Halt();
11 }
```

- consoleIO_test2.c

```
1 #include "syscall.h"
2
3 int
4 main()
5 {
6     int n;
7     for (n=15;n<=19;n++){
8
9         PrintInt(n);
10    }
11    Halt();
12 }
```

MP – System call

- Part II: File I/O system call
- fileIO_test1.c

```
1 #include "syscall.h"
2
3 int main(void)
4 {
5     char test[] = "abcdefghijklmnopqrstuvwxyz";
6     int success = Create("file1.test");
7     OpenFileId fid;
8     int i;
9     if (success != 1) Fail("Failed on creating file");
10    fid = Open("file1.test");
11    if (fid <= 0) Fail("Failed on opening file");
12    for (i = 0; i < 26; ++i) {
13        int count = Write(test + i, 1, fid);
14        if (count != 1) Fail("Failed on writing file");
15    }
16    success = Close(fid);
17    if (success != 1) Fail("Failed on closing file");
18    Halt();
19 }
```

- fileIO_test2.c

```
1 #include "syscall.h"
2
3 int main(void)
4 {
5     // you should run fileIO_test1 first before running this one
6     char test[26];
7     char check[] = "abcdefghijklmnopqrstuvwxyz";
8     OpenFileId fid;
9     int count, success, i;
10    fid = Open("file1.test");
11    if (fid <= 0) MSG("Failed on opening file");
12    count = Read(test, 26, fid);
13    if (count != 26) MSG("Failed on reading file");
14    success = Close(fid);
15    if (success != 1) MSG("Failed on closing file");
16    for (i = 0; i < 26; ++i) {
17        if (test[i] != check[i]) MSG("Failed: reading wrong result");
18    }
19    MSG("Passed! ^_^");
20    Halt();
21 }
```

Assignment – Part I

- For part I, You have to implement `PrintInt(int number)` system call
- You **should NOT** use standard library IO functions in any part of Nachos
 - e.g. `putchar()`, `printf()`, `cout`, ...
- result should be like this:

```
[test@lsalab test]$ ../build.linux/nachos -e consoleIO_test1
consoleIO_test1
9
8
7
6
Machine halting!

This is halt
Ticks: total 669, idle 400, system 180, user 89
Disk I/O: reads 0, writes 0
Console I/O: reads 0, writes 4
Paging: faults 0
Network I/O: packets received 0, sent 0
```

Assignment – Part II

- For second part, you have to implement four file I/O system call
 - `OpenFileId Open(char *name);`
 - open a file with the name and return its fileId
 - `int Write(char *buffer, int size, OpenFileId id);`
 - write "size" characters from buffer into the file
 - return number of characters actually written to the file
 - `int Read(char *buffer, int size, OpenFileId id);`
 - read "size" characters from the file and copy them into buffer
 - return number of characters actually read from the file
 - `int Close(OpenFileId id);`
 - return 1 if successfully close the file, 0 otherwise

Assignment – Part II

- You **should NOT** use standard library IO functions in any part of Nachos
 - e.g. open(), close(), read(), write(), fread(), fwrite(), ...
- A successful run of fileIO_test1 will generate an file “file1.test”

```
-rw-rw-r-- 1 test test    26 Sep 24 23:47 file1.test
-rw-rw-r-- 1 test test   980 Sep 24 23:46 fileIO_test1
-rw-rw-r-- 1 test test   478 Sep 24 23:47 fileIO_test1.c
-rwxrwxr-x 1 test test  4712 Sep 24 23:46 fileIO_test1.coff
-rw-rw-r-- 1 test test  1640 Sep 24 23:46 fileIO_test1.o
-rw-rw-r-- 1 test test   980 Sep 24 23:46 fileIO_test2
-rw-rw-r-- 1 test test   540 Sep 24 23:48 fileIO_test2.c
-rwxrwxr-x 1 test test  4712 Sep 24 23:46 fileIO_test2.coff
-rw-rw-r-- 1 test test  1592 Sep 24 23:46 fileIO_test2.o
```

- content in file1.test

```
1 a b c d e f g h i j k l m n o p q r s t u v w x y z
```


Assignment – Part II

- A successful run of fileIO_test2 should look like this

```
[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
```

- You will get a ^_^ mark if you pass the test
- Disk I/O remains 0 , since we are now using LINUX backed file system
- We will implement it in MP4, coming soooooooooon~

Grading

- Work item1: Implementation – 60%
 - Demo (**on course server**)
 - console I/O – 20%, file I/O – 40%
- Work item 2: tracing system call – 25%
 - Explain how system calls go through NachOS in detail (ex. Halt() , Create())
 - Included in the report
- Work item 3: report – 15%
 - Explain your work (modifications of the code, team member contribution, ...)
- **Deadline: 2017/10/16(Monday) 08:00, penalty for late submission**
- **You can discuss, but do not copy. 0 will be given to cheaters.**

Hint – part I

- Trace how **Halt()** system call works, this will help you a lot.
- **Do not** trace Add() system call, this is not a console IO system call
- Interrupt is important!
- Files to modify
 - userprog/syscall.h, exception.cc, ksyscall.h, synchconsole
 - machine/console, interrupt
 - test/start.S
 - threads/kernel

Hint – part II

- Trace how **Create()** system call works, this will help you a lot.
- **Return value** is important!!!
- Files to modify
 - userprog/syscall.h, exception.cc, ksyscall.h
 - machine/interrupt
 - filesys/filesys, openfile
 - threads/kernel
- **You can not directly use the pointer of a user program**, remember to translate it before using them in kernel!!!

FAQ

- Q1: cannot login or cannot connect to server
- A1: Please contact TA, do not try to guess your password by brute force.
(We may ban your IP for safety issues)
- Q2: "xxx: permission denied"
- A2: change the target program permission via command chmod
 - e.g. >chmod 775 ./my_program

FAQ

- Q3: I modified code in nachos, but nothing seems changed?
- A3: Be sure you do not get any errors during make. (You can ignore warning messages) **And always remember to “make” after you modify anything in Nachos.**
- Q4. How do I create my testing program?
- A4. Please modify the provided testing files and make, or you can learn how to write/modify a makefile. **Do not compile your testing code with gcc, this will make the binary unrunnable on Nachos!!!**

FAQ

- Q5: My process got stuck, how do I escape?
- A5: press [ctrl] + C to kill the process.
- Q6: Do TAs help debug my Nachos?
- A6: **No**, we only discuss concept with you. You can still ask questions about weird error messages.
- If you have other questions, feel free to ask on iLMS.

Reference

- Linux command
 - http://linux.vbird.org/linux_basic/0220filemanager.php
- vim command
 - http://linux.vbird.org/linux_basic/0310vi.php
- NachOS
 - <http://homes.cs.washington.edu/~tom/nachos/>