NachOS-MP1

Lecturer: Jerry Chou
TA: Cake, Vincent
National Tsing Hua University

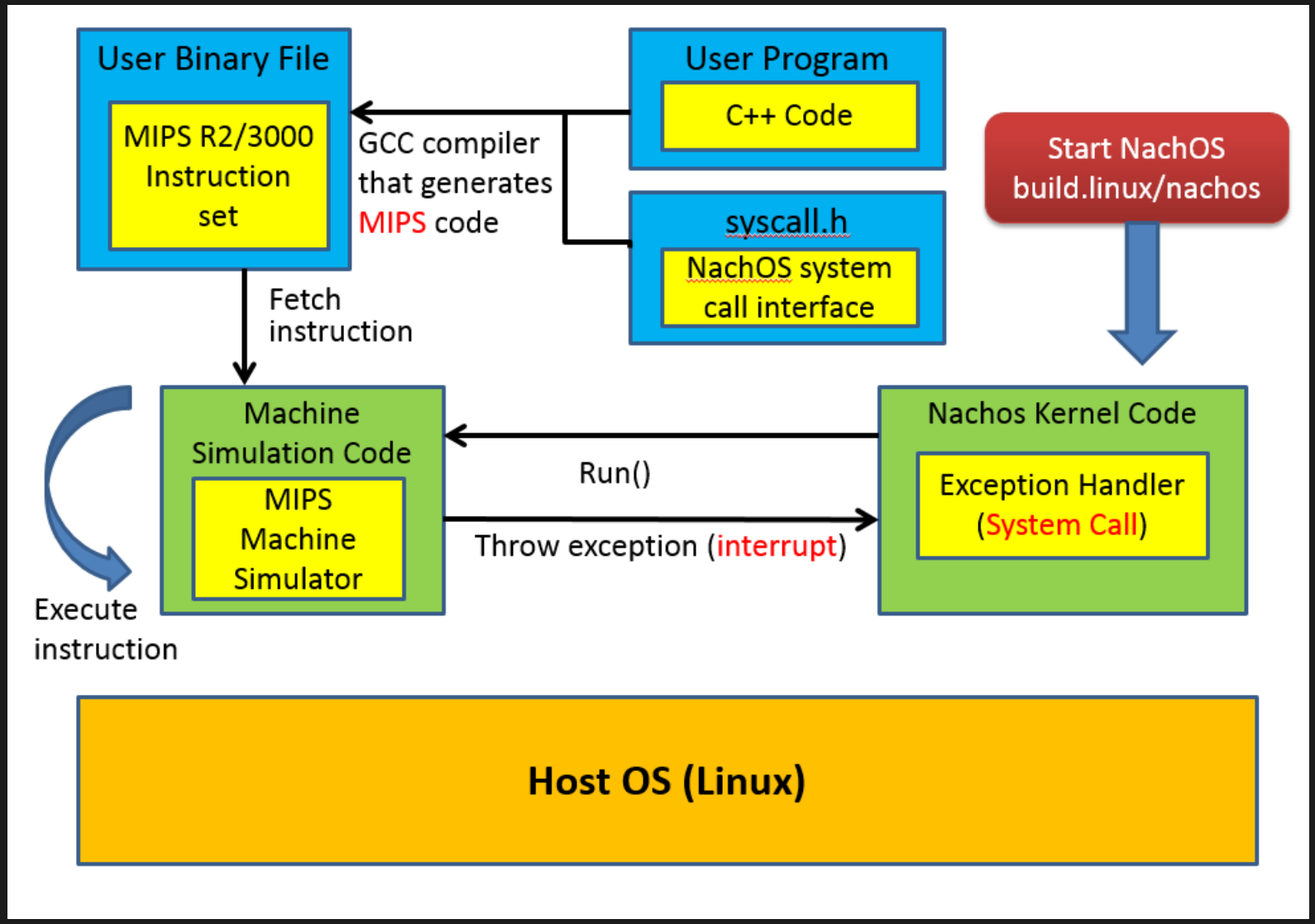
Outline

- 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



Installation of Nachos

- IP address: 140.114.78.243 port:22 (ssh)
  - Account: 2015osteam + your teamID (e.g. 2015osteam01, 2015osteam15, …)
  - Passwd: You will be ask to set up your password once you login
  - contact TA if you have problem logging in
- Installation
  - `>cp -r /home/os2015/shared/NachOS-4.0_MP1 .`
  - `>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

```c
#include "syscall.h"

int main(void)
{
    char test[] = "abcdefghijklmnopqrstuvwxyz";
    int success = Create("file1.test");
    OpenFileId fid;
    int i;
    if (success != 1) Fail("Failed on creating file");
    fid = Open("file1.test");
    if (fid <= 0) Fail("Failed on opening file");
    for (i = 0; i < 26; ++i) {
        int count = Write(test + i, 1, fid);
        if (count != 1) Fail("Failed on writing file");
    }
    success = Close(fid);
    if (success != 1) Fail("Failed on closing file");
    Halt();
}
```

- fileIO_test2.c

```c
#include "syscall.h"

int main(void)
{
    // you should run fileIO_test1 first before running this one
    char test[26];
    char check[] = "abcdefghijklmnopqrstuvwxyz";
    OpenFileId fid;
    int count, success, i;
    fid = Open("file1.test");
    if (fid <= 0) MSG("Failed on opening file");
    count = Read(test, 26, fid);
    if (count != 26) MSG("Failed on reading file");
    success = Close(fid);
    if (success != 1) MSG("Failed on closing file");
    for (i = 0; i < 26; ++i) {
        if (test[i] != check[i]) MSG("Failed: reading wrong result");
    }
    MSG("Passed! ^_^");
    Halt();
}
```

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 abcdefghijklmnopqrstuvwxyz
```

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 soooooooon~

Grading

- Program correctness – Demo (on Server)
  - console I/O – 20%, file I/O – 40%
- Work item 1: tracing system call – 25%
  - save it as a PDF
  - explain how system calls go through NachOS in detail
- Work item 2: report – 15%
  - explain your work (modifications of the code, team member contribution, …)
- Deadline: 2015/10/18(Sunday) 23:59, 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/