國立陽明交通大學
NATIONAL YANG MING CHIAO TUNG UNIVERSITY

Institute of Artificial Intelligence Innovation
Department of Computer Science

Operating System

Homework 01: System Call

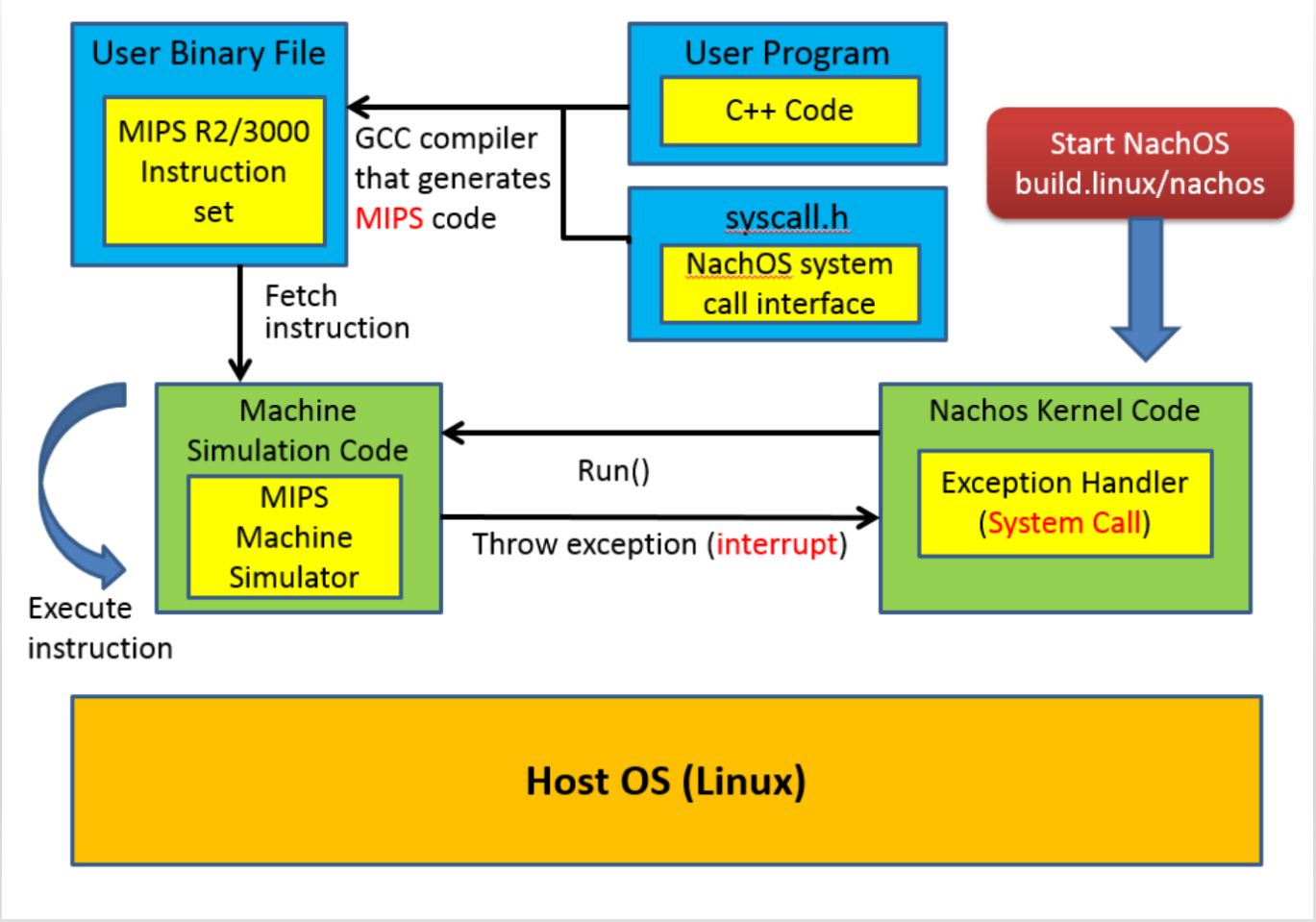
Shuo-Han Chen (陳碩漢),
shch@nycu.edu.tw

R13:20 - 16:20 ED305

Goal

- Understand how to work in Linux Environment
- Understand how system calls are implemented by OS
- Understand the difference between user mode and kernel mode

Introduction

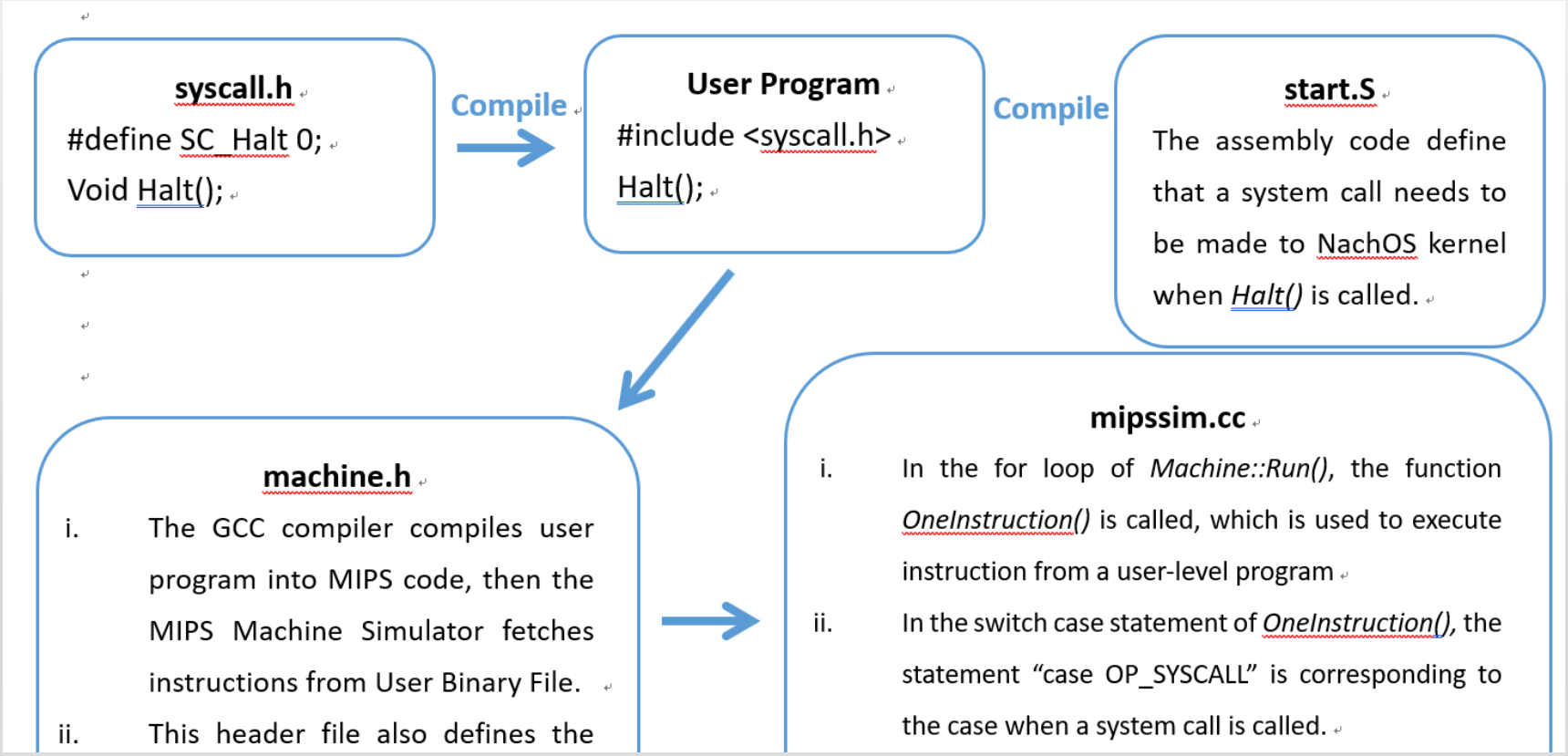


Part I

- Trace how Halt() system call works
  - Explain how system calls go through NachOS in details
- Trace how Create() system call works
  - Explain the basic operations and data structure in a file system
- Trace the Makefile in code/test/Makefile to understand how test files are compiled
- Files to look into
  - userprog/syscall.h, exception.cc, ksyscall.h, synchconsole
  - machine/mipssim, interrupt
  - filesys/openfile, filesys
  - test/start.s, halt.c, Makefile
  - threads/kernel
- You should include two things in the report
  - Flow chart of system call (Halt, Create)
  - Tracing details of code (Halt, Create, Makefile)

Flow Chart of System Call

- It should look like …



Tracing Details of Code

- Just paste the code with nice arrangement
- Don’t paste the whole file, just the part that will be used

1. machine.h

```
void Run();
```

2. mipssim.cc

```
Machine::Run();
```

```
for (;;) {
        OneInstruction(instr);
        kernel->interrupt->OneTick();
        if (singleStep && (runUntilTime <= kernel->stats->totalTicks))
            Debugger();
    }
```

3. mipssim.cc

```
void Machine::OneInstruction(Instruction *instr)
```

Part II

- Implement a console I/O system call

```
void PrintInt (int number)
        // Output the number and a line separator to the console.
```

- Implement four file I/O system call

```
OpenFileId Open(char *name);
        // Open a file with the name, and returns its corresponding OpenFileId.
        // Return -1 if open fails
int Write(char *buffer, int size, OpenFileId id);
        // Write “size” characters from buffer into the file
        // Returns number of characters actually written to the file
        // If attempt writing to an invalid id, return -1
int Read(char *buffer, int size, OpenFileId id);
        // Read “size” characters from file into the buffer
        // Returns number of characters actually read from the file
        // If attempt reading from an invalid id, return -1
int Close(OpenFileId id);
        // Close the file with id
        // Return 1 if successfully close the file, 0 otherwise
```

Requirement

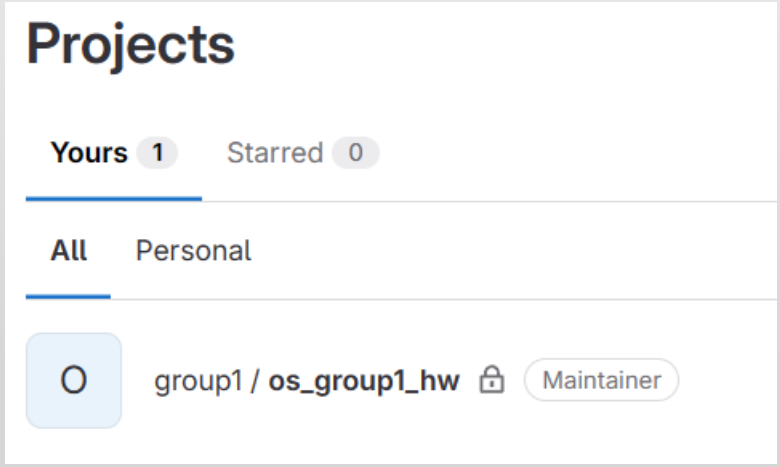
- All your implemens should not use any IO functions from standard libraries (e.g. printf(), cout, fopen(), fwrite(), write(), etc.).
- Must handle invalid file open requests, including the non-existent file, exceeding opened file limit (at most 20 files)
- Must handle invalid file read, write, close requests, including invalid id

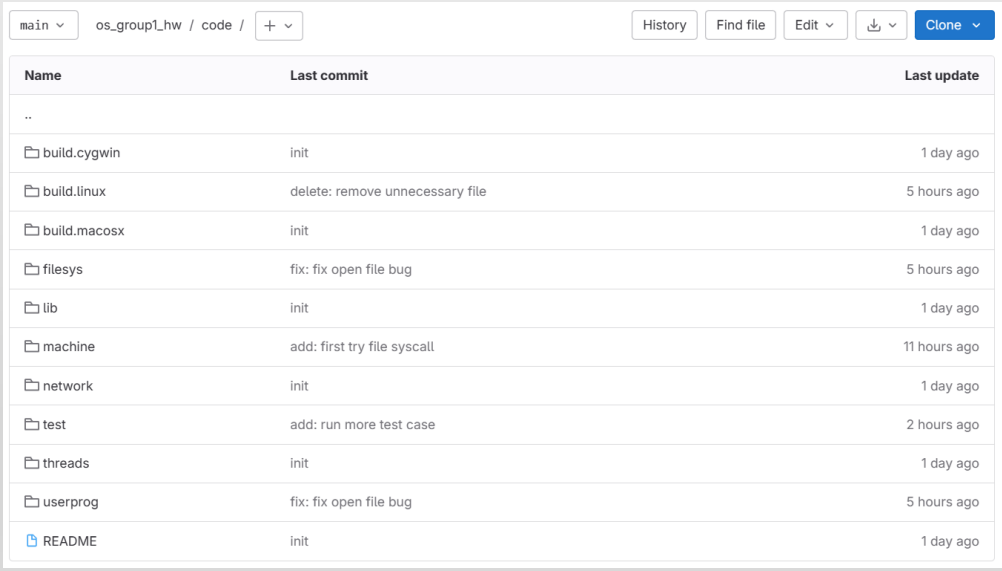
Hint

- 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/
- You can run consoleIO_test1.c, consoleIO_test2.c to verify consoleIO part
- You can run fileIO_test1.c, fileIO_test2.c to verify fileIO part

Gitlab Nachos Repository

- Gitlab Link: https://css-nachos.hopto.org/gitlab/
- After logging into your gitlab account, you should see your group project
- Your Nachos file will already be inside the project





Jenkins Job

- Jenkins Link: https://css-nachos.hopto.org/jenkins/
  - Account : studentID
  - Password : !23456789
- You should modify your default password
- After logging into your jenkins account, you should see your group jobs

How to run Nachos? (Recommended method)

1. modify nachos
2. commit & push to gitlab

3. push event will trigger Jenkins

4. trigger os_group_hw job first

os_group_hw job

1. compile your nachos
    1. go to code/build.linux
    2. make depend
    3. make clean
    4. make
2. run students custom shell
    1. go to code/test
    2. bash os_students.sh

os_students.sh 235 B

```
1  make clean
2  make
3  ../build.linux/nachos -e halt
```

// you can only customize your scripts inside this .sh file

5. after finished hw job, it will trigger os_group_ta job

os_group_ta job

1. compile your nachos
2. run secret test case to evaluate your implementation

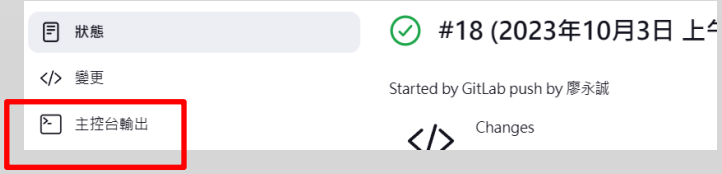
Jenkins Description

- You can view the output of your os_students.sh in the Jenkins os_group_hw console

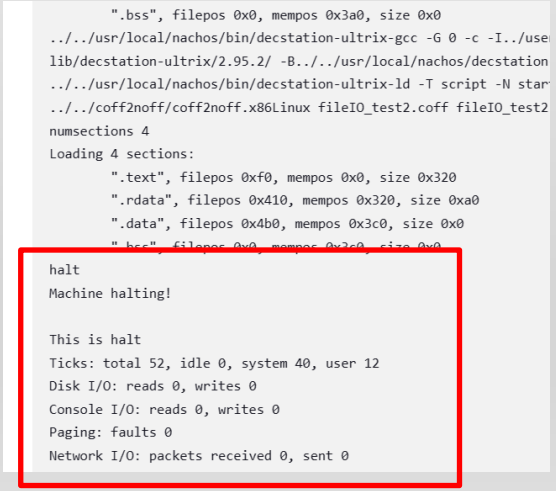
1. click your job

2. click specific build you want to see

3. click console output



4. scroll down to bottom you should see the output



// output from "../build.linux/nachos –e halt"

Jenkins os_group_ta Description

- You will have 8 test cases, and each test case is 9% of the total grade
- In print test, you should verify that the number of Console I/O writes is correct

```
==================================
Running the test: mp1_print_test1
==================================
65
mp1_print_test1
result is 65
Machine halting!

This is halt
Ticks: total 197, idle 100, system 70, user 27
Disk I/O: reads 0, writes 0
Console I/O: reads 0, writes 1
```

```
==================================
Running the test: mp1_print_test2
==================================
9
10
11
12
mp1_print_test2
Machine halting!

This is halt
Ticks: total 679, idle 400, system 180, user 99
Disk I/O: reads 0, writes 0
Console I/O: reads 0, writes 4
```

- In the file test, you should verify that your output includes the string “Passed Test!” for each test

```
==================================
Running the test: mp1_file_test1
==================================
mp1_file_test1
Passed Test!
Machine halting!
```

Run Locally

- If you want run nachos locally, please follow the steps below

1. install new virtual machine (Only well-tested on Ubuntu 22.04 LTS 64bits)
2. git clone your group project
3. install compile dependency
    1) sudo apt-get install build-essential
    2) sudo dpkg --add-architecture i386
    3) sudo apt-get install csh
    4) sudo apt-get update
    5) sudo apt-get dist-upgrade
    6) sudo apt-get install gcc-multilib g++-multilib
    7) sudo apt-get install lib32ncurses5-dev lib32z1
    8) sudo apt-get install zlib1g:i386 libstdc++6:i386
    9) sudo apt-get install libc6:i386 libncurses5:i386
    10) sudo apt-get install libgcc1:i386 libstdc++5:i386

Run Locally (cont'd)

- modify code/build.linux/Makefile
- compile nachos
  - cd code/build.linux/
  - make depend
  - make clean
  - make
- compile test case
  - cd code/test/
  - make clean
  - make
- test output
  - ../build.linux/nachos -e halt

Makefile

```
// the other line ...
CPP=/lib/cpp
CC = g++ -m32 -Wno-deprecated
LD = g++ -m32 -Wno-deprecated
AS = as --32
RM = /bin/rm
// the other line ...
```

Grading

- PartI (Trace System call) – 25%
- PartII (Implement System call)– 72%
  - Console I/O system call – 24%
  - File I/O system call – 48%
- Report Format – 3%

- Deadline: 10/26

Report Format

- Please follow the word file to form your report for HW01
- Format guide
  - Content format: should be set with 12pt front,16pt row height, and align to the left.
  - Caption format: 18pt and Bold font.
  - Font format: Times New Roman.
  - Figure: center with single line row height.
  - Change the title to your student ID and name in Chinese.
  - Upload pdf file with the file name format : OS_HW01_GROUP_X.pdf (change X to your group ID)

Reminder

- 0 will given to cheaters. Do not copy & paste!
  - TA will check your repository
- Feel free to ask TA questions
  - Teams Message(Recommended): 廖永誠
  - Email: yongchengliaw.ii12@nycu.edu.tw

Q & A

Thank you for your attention