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

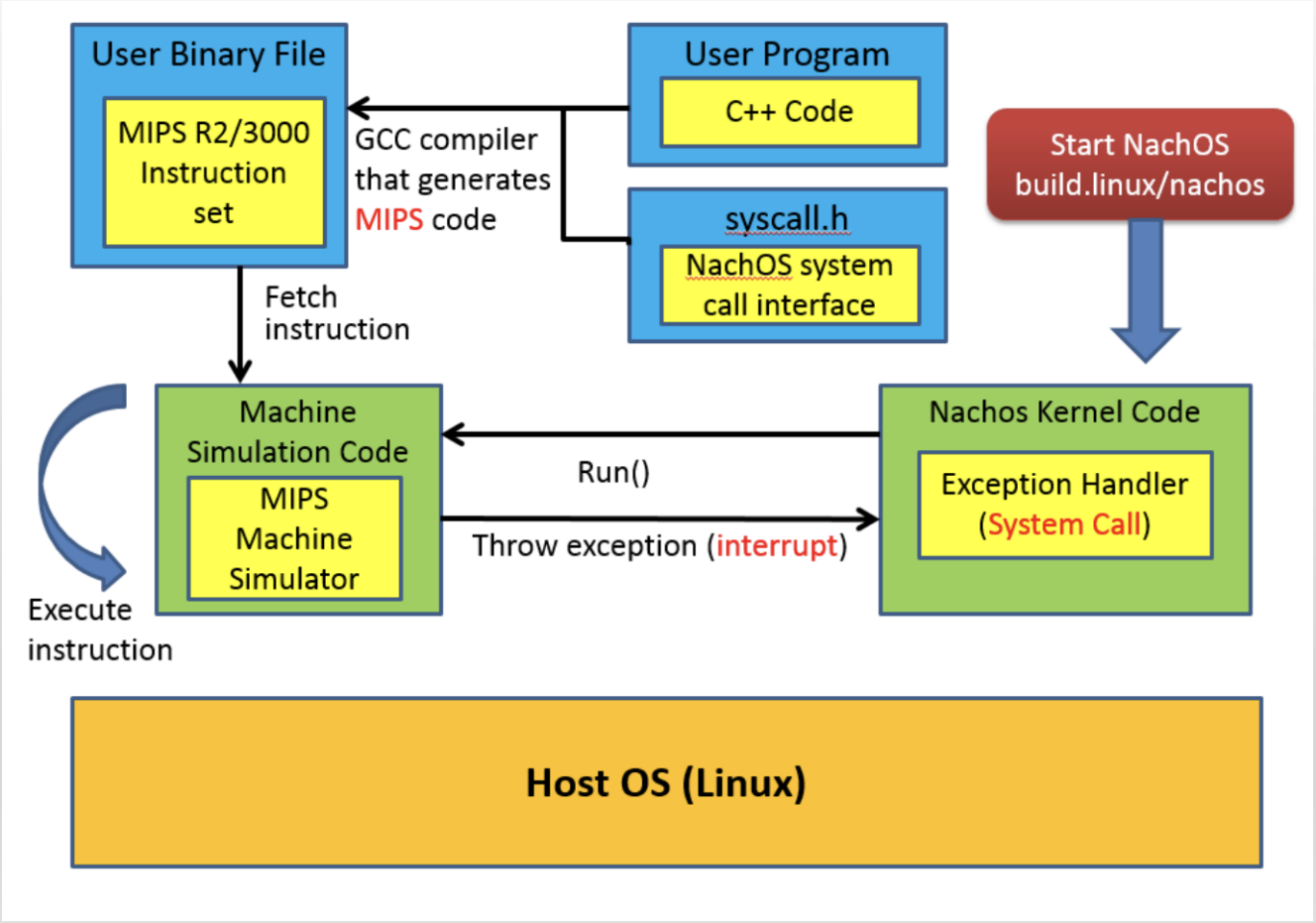
Shuo-Han Chen (陳碩漢),
shch@nycu.edu.tw

R13:20 - 16:20 ED201

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

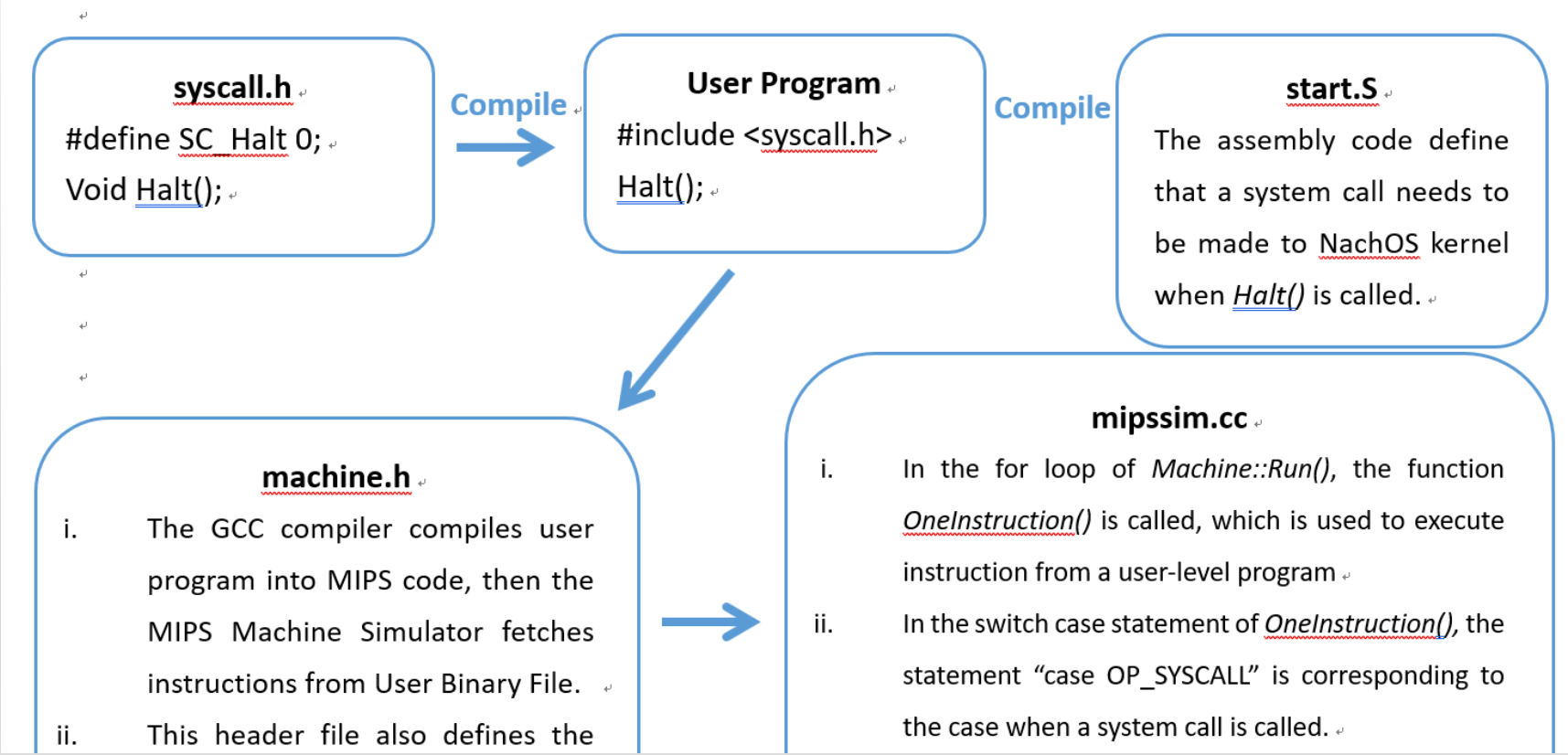


Part I

- 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, console
  - 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

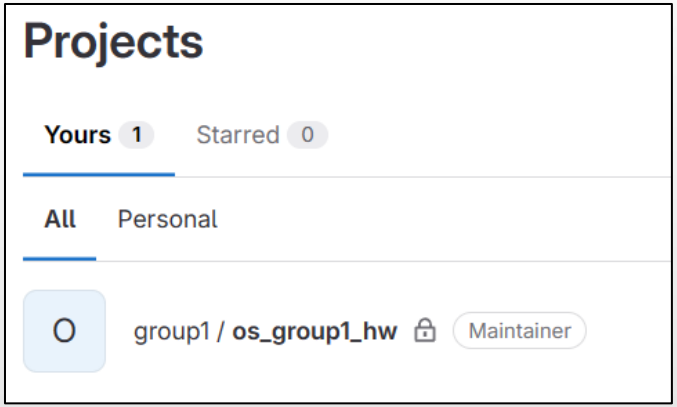
- All your implemens should not use any IO functions from standard libraries (e.g. printf(), cout, fopen(), fwrite(), write(), etc.).
- Must handle invalid file open requests, including the non-existent file, exceeding opened file limit (at most 20 files)
- Must handle invalid file read, write, close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

Hint

- Do not change or remove the flag -DFILESYS_STUB in the Makefile
  - We use the stub file system for this homework
  - Path: build.linux/Makefile
- Default test case is under /code/test
  - Test for ConsoleIO
    - consoleIO_test1.c
    - consoleIO_test2.c
  - Test for FileIO
    - fileIO_test1.c
    - fileIO_test2.c

Gitlab Nachos Repository

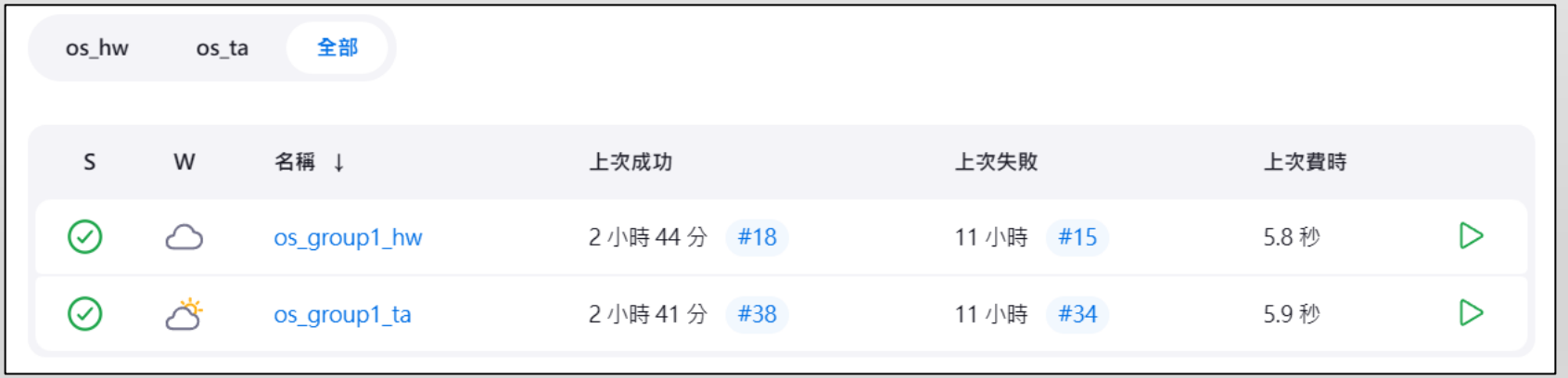
- Gitlab Link: https://css-nachos.hopto.org/gitlab/
  - Account : studentID
  - Password : TBD
  - You should modify your default password



- 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 : TBD
  - You should modify your default password
- you should see your group jobs in Jenkins after you login



How to run Nachos? (Recommended method)

1. clone the project
2. modify nachos
3. commit & push to gitlab

4. push event will trigger Jenkins

5. trigger os_group_hw job first

os_group_hw job

1. clone the branch you pushed
2. compile your nachos
    1. go to code/build.linux
    2. make depend
    3. make clean
    4. make
3. 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
```

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

os_group_ta job

1. clone the main branch
2. compile your nachos
3. run secret test case to evaluate your implementation

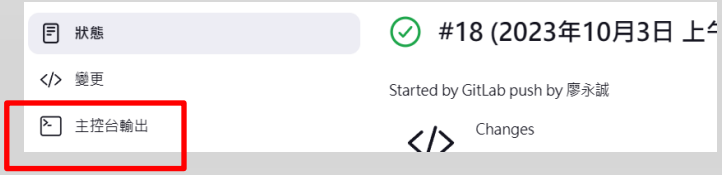
// you can only customize your scripts inside this .sh file

Jenkins Description

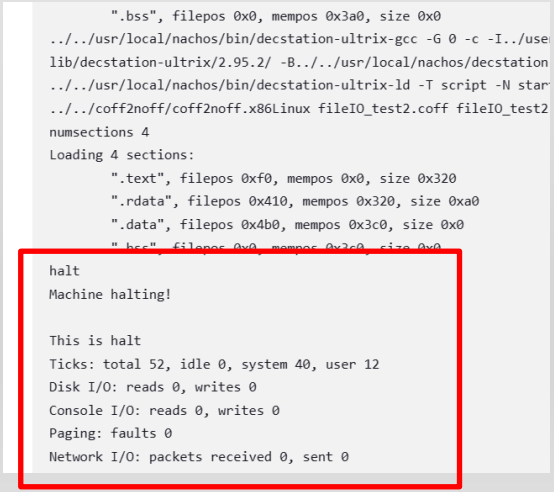
- You can view the output of your os_students.sh in the Jenkins os_group_hw console

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 dpkg --add-architecture i386
   2) sudo apt install build-essential
   3) sudo apt install gcc-multilib g++-multilib

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 ...

- **Solution for error message "make : ../../usr/local/nachos/bin/decstation-ultrix-gcc: Permission denied"**
  - cd NachOS-4.0/
  - chmod -R 777 ./usr ./coff2noff

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: 10/5 23:59

Report Format

- Please follow the word file to form your report for HW01
- Format guide
  - Content format: 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.
  - Upload pdf file with the file name format :
    - OS_HW01_GROUP_X.pdf (change X to your group ID)

Reminder

- The homework is considered passed **only if the TA job** passes
- 0 will given to cheaters. Do not copy & paste!
  - TA will check your repository
- Feel free to ask TA questions
  - The TA will only assist you with GitLab, Jenkins environment problems, or any issues related to homework requirements.
  - The TA will not help you debug your code.
  - Teams Message(Recommended): 蔡宇翔
  - Email: yu.ii13@nycu.edu.tw

S	W	名稱 ↓	上次成功	上次失敗	上次費時
✓	☁	os_group1_hw	3 小時 6 分 #18	11 小時 #15	5.8 秒
✓	☁	os_group1_ta	3 小時 4 分 #38	11 小時 #34	5.9 秒

Q & A

Thank you for your attention