Homework Tutorial

Speaker: Wei-Xiang Wang

M10715010@mail.ntust.edu.tw

Requirement

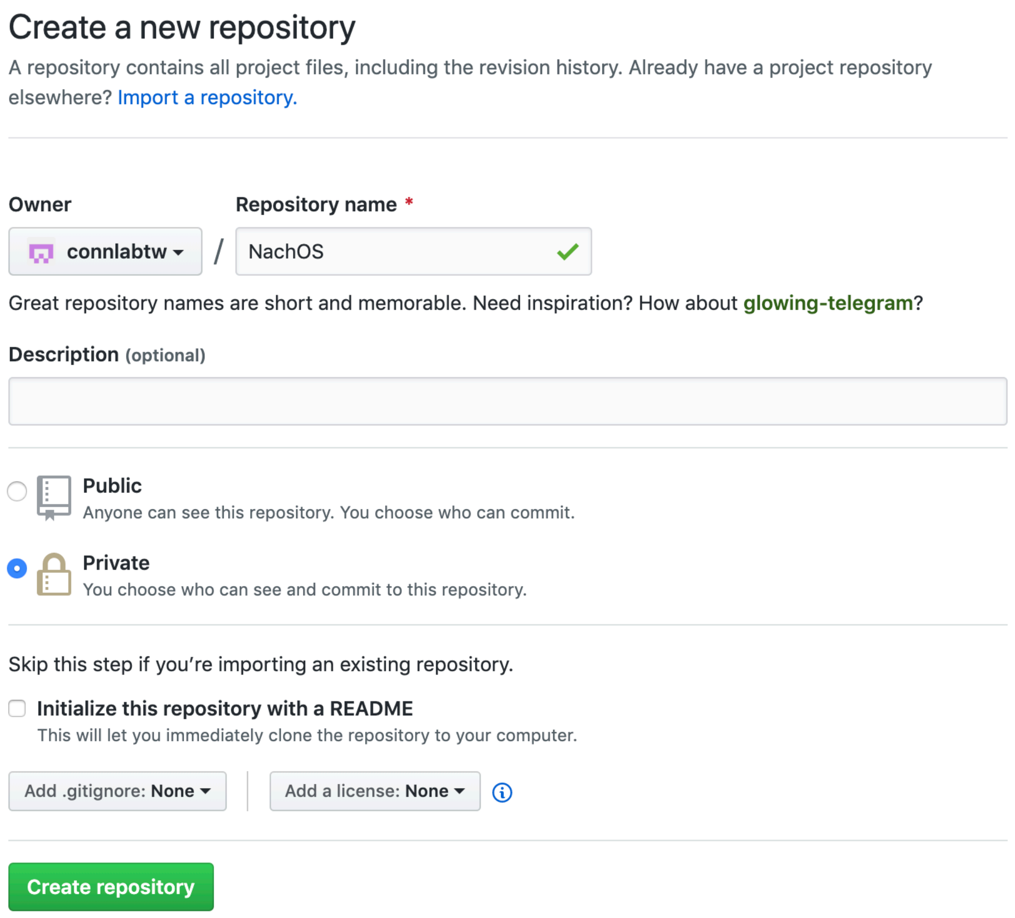
- Github account
- Fill in google form with student No. and github ID
  - https://forms.gle/Gd2m4xSUEXNKWzv58

Create a repository from github

- Repository name
  - NachOS

Attention to uppercase

- Private



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Owner: connlabtw / Repository name *: NachOS ✓

Great repository names are short and memorable. Need inspiration? How about glowing-telegram?

Description (optional)

○ Public
Anyone can see this repository. You choose who can commit.

● Private
You choose who can see and commit to this repository.

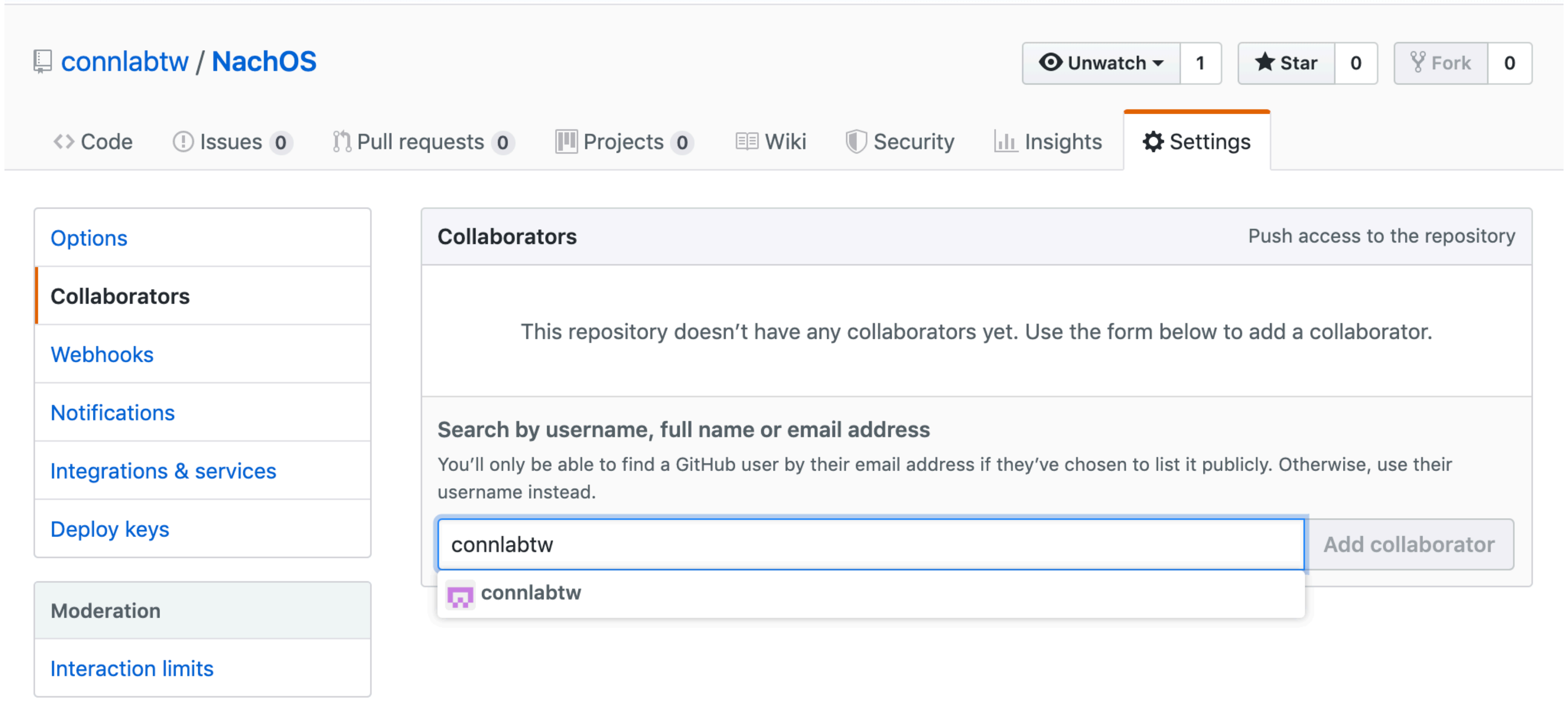
Skip this step if you're importing an existing repository.

☐ Initialize this repository with a README
This will let you immediately clone the repository to your computer.

Add .gitignore: None | Add a license: None

Collaborator

› Add *connlabtw* to your repository collaborator


connlabtw / NachOS

Collaborators — Push access to the repository

This repository doesn't have any collaborators yet. Use the form below to add a collaborator.

Search by username, full name or email address

You'll only be able to find a GitHub user by their email address if they've chosen to list it publicly. Otherwise, use their username instead.

connlabtw — Add collaborator

connlabtw

NachOS Introduction

- NachOS OS runs on the host machine
- NachOS user processes run on machine simulator
- Real OS kernel can be preempted anywhere the interrupts are enabled
  - NachOS time is simulated vs. Real OS time is continue
  - Interrupts happen only in places where simulated time gets advanced. When simulated time advances, NachOS will check for interrupts

NachOS Requirement

- Programing Language : C and C++
- Operating System: ubuntu (32/64 bit)
  - Recommended ubuntu 14.04 and older (32 bit)
  - If you want to use 64 bit, you need to patch
- Virtualization (Optional) :
  - Vmware workstation or Virtualbox
- MIPS cross-compiler:
  - NachOS user programs need it
- System Architecture:
  - Shown in the right figure

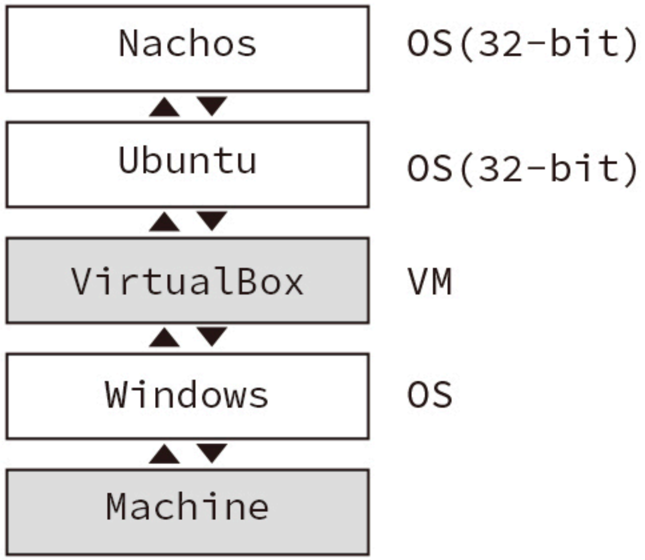
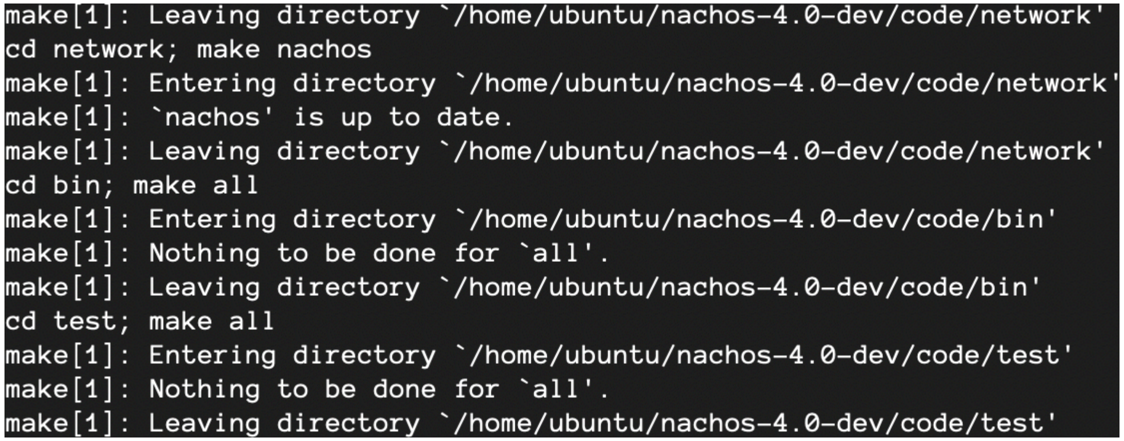


Figure reference: jeffprogrammer blog

NachOS Installation

```
$ sudo apt install git csh g++
$ git clone https://github.com/connlabtw/NachOS.git
$ cd NachOS
$ sudo cp -r usr /
$ cd code
$ make
```



```
make[1]: Leaving directory `/home/ubuntu/nachos-4.0-dev/code/network'
cd network; make nachos
make[1]: Entering directory `/home/ubuntu/nachos-4.0-dev/code/network'
make[1]: `nachos' is up to date.
make[1]: Leaving directory `/home/ubuntu/nachos-4.0-dev/code/network'
cd bin; make all
make[1]: Entering directory `/home/ubuntu/nachos-4.0-dev/code/bin'
make[1]: Nothing to be done for `all'.
make[1]: Leaving directory `/home/ubuntu/nachos-4.0-dev/code/bin'
cd test; make all
make[1]: Entering directory `/home/ubuntu/nachos-4.0-dev/code/test'
make[1]: Nothing to be done for `all'.
make[1]: Leaving directory `/home/ubuntu/nachos-4.0-dev/code/test'
```

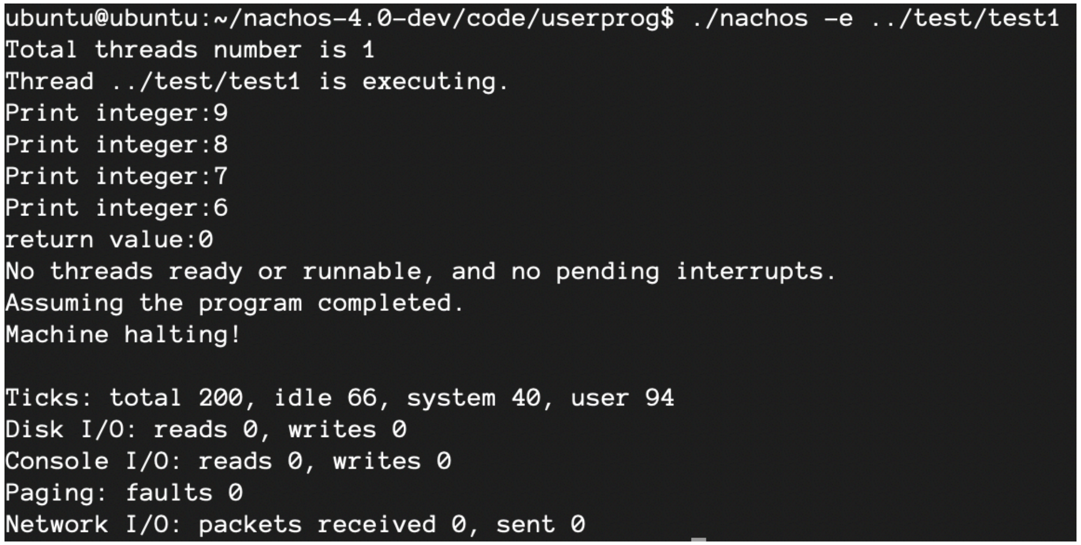
NachOS Testing

$ cd userprog

$ pwd

/home/ubuntu/NachOS/code/userprog

$ ./nachos –e ../test/test1



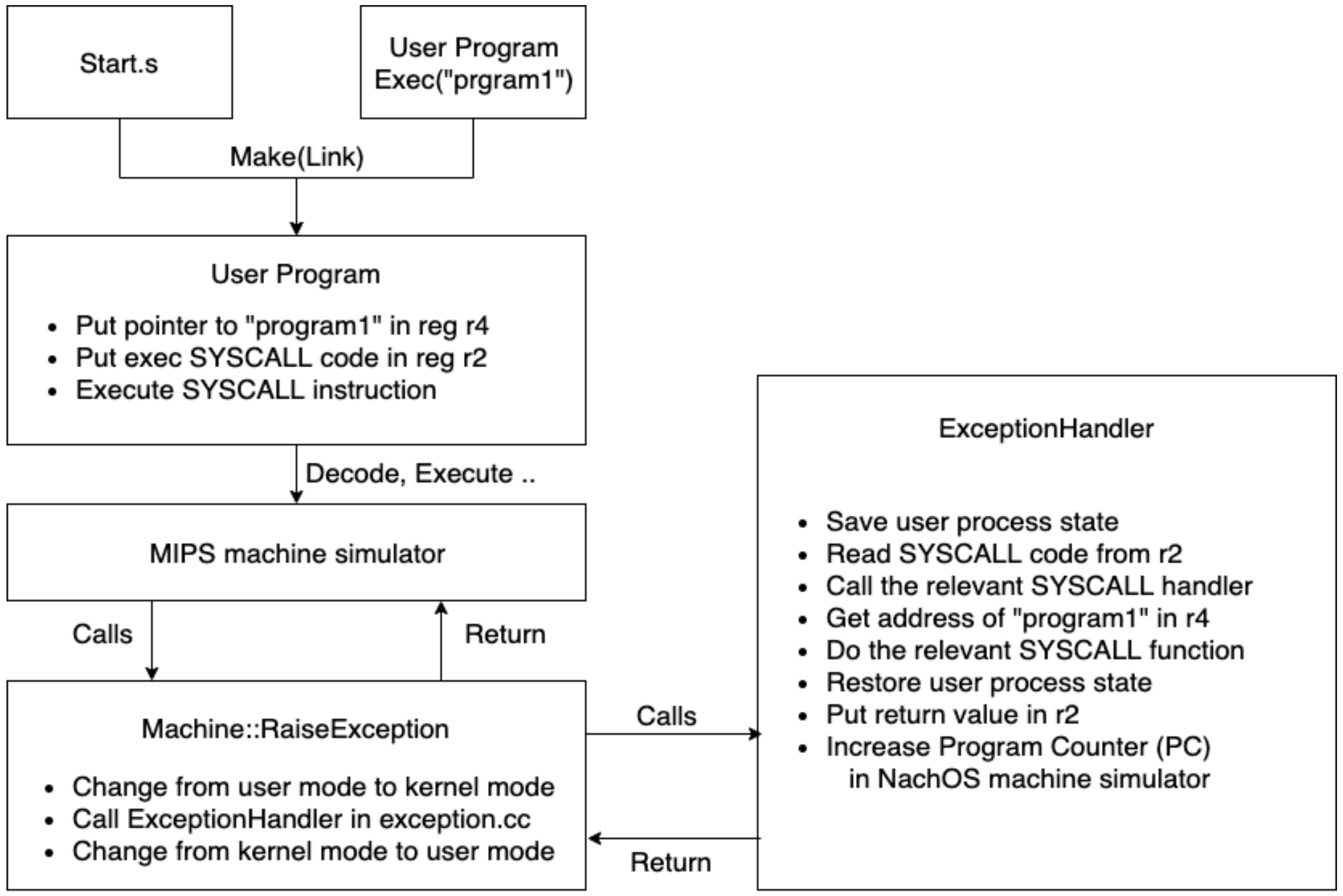
```
ubuntu@ubuntu:~/nachos-4.0-dev/code/userprog$ ./nachos -e ../test/test1
Total threads number is 1
Thread ../test/test1 is executing.
Print integer:9
Print integer:8
Print integer:7
Print integer:6
return value:0
No threads ready or runnable, and no pending interrupts.
Assuming the program completed.
Machine halting!

Ticks: total 200, idle 66, system 40, user 94
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
```

NachOS Startup

1. Initialize simulated hardware
   - e.g. timer, console, disk
2. Initialize kernel data structures
   - e.g. proc table, scheduler, file system
3. Load initial process from disk, create its address space and thread object
4. Setup machine simulator to run process by initializing registers and setting the machine page table to point to process page table
5. Run the process: machine->Run()

NachOS System Call


Start.s
User Program
Exec("prgram1")
Make(Link)
User Program
Put pointer to "program1" in reg r4
Put exec SYSCALL code in reg r2
Execute SYSCALL instruction
Decode, Execute ..
MIPS machine simulator
Calls
Return
Machine::RaiseException
Change from user mode to kernel mode
Call ExceptionHandler in exception.cc
Change from kernel mode to user mode
Calls
Return
ExceptionHandler
Save user process state
Read SYSCALL code from r2
Call the relevant SYSCALL handler
Get address of "program1" in r4
Do the relevant SYSCALL function
Restore user process state
Put return value in r2
Increase Program Counter (PC) in NachOS machine simulator

NachOS System Call

- There is no save & restore user process state because OS and user process run on different machines
- Real OS would have save & restore process state since OS and process run on same machine

NachOS Testing Tips

- Test programs are in C. NachOS is in C++
- Can use debug flags to print things in kernel code. e.g. DEBUG(dbgSysCall, "System Call: Exit status=" << status);
- Can make your own debug flags. See debug.h
- Can use "./nachos -d flag –e ../test/test1" to print debug message you want to. (flag see debug.h)
- Don't put a big array on stack; otherwise it fill overflow.
- To shut down NachOS, call nachOS/code/test/Halt

Example: Implement SYSCALL

› Add new system call define to /code/userprog/syscall.h (e.g. #define SC_Example 13)

› Add *void Example(int number);*

  to code/userprog/syscall.h

Example: Implement SYSCALL (Cont.)

› Implement exception handler in code/userprog/exception.cc

```
case SC_Example:
        val=kernel->machine->ReadRegister(4);
        cout << "Example value:" <<val << endl;
        return;
```

› Add assembly code to /code/test/start.s

```
    .globl Example
    .ent Example
Example:
    addiu       $2,$0,SC_Example
    syscall
    j       $31
    .end    Example
```

Example: Make user programs

- Create a example.c in code/test/

```
#include "syscall.h"
main() {
            int n;
            for (n=1;n<5;n++)
                Example(n);
}
```

- Edit */code/test/Makefile*

```
…
all: halt shell matmult sort test1 example
…
example: example.o start.o
    $(LD) $(LDFLAGS) start.o example.o -o example.coff
    ../bin/coff2noff example.coff example
…
```

- Recompile NachOS
- ./nachos -e ../test/example

HW 1: Implement SYSCALL sleep

- You have to implement “WaitUntil” function in code/threads/alarm.cc
- It already defined in /code/threads/alarm.h
- Trace the following files
  - code/threads/alarm.cc
  - code/threads/alarm.h
  - code/threads/scheduler.cc (ReadyToRun)
  - code/machine/time.cc
  - code/machine/interrupt.cc
  - code/machine/interrupt.h
  - code/userprog/exception.cc

CONNECTIVITY LAB

HW 1: Implement SYSCALL sleep

- A hardware timer generates a CPU interrupt every X milliseconds.
- Trigger interrupt by NachOS Class Alarm
- Design a method to count sleep value
- Put thread into ready queue after process has waken
  - Tip: kernel->scheduler->ReadyToRun(*thread*);
- You have to ensure process won't be preempted when it runs in "WaitUntil" function
  - Tip: kernel->interrupt->SetLevel(IntStatus);

Hand in source code & report

- Source Code
  - Push it to your GitHub
- Report
  - The idea you take to implement the problem in NachOS
  - Some important codes and comments
  - Experiment result (Screenshot)
  - Saved as [Student ID]_HW1_report.pdf
    - e.g. M10715010_HW1_report.pdf
  - Upload report to moodle