Operating System - NachOS Project Introduction

教授：周立德

助教：楊宏昌

分散式系統實驗室 工五館 A312 分機：35231

Email: cyht@dslab.csie.ncu.edu.tw

NachOS

- Not Another Completely Heuristic Operating System
- Written by Tom Anderson and his students at UC Berkeley in 1992
  http://www.cs.washington.edu/homes/tom/nachos/

What is NachOS

- An educational OS used to
  - Teach monolithic kernel design and implementation
  - Do experiments, you can (and will) modify and extend it
- Fact
  - Real hardware is difficult to handle
  - May break if handled wrong

What is NachOS

- Approach
  - Use a virtual MIPS machine
  - Provide some basic OS elements
- Includes many facets of a real OS:
  - Threads
  - Interrupts
  - Virtual Memory
  - I/O driven by interrupts

What is NachOS

- NachOS also contains some hardware simulation
  - MIPS processor
    - Can handle MIPS code in standard COFF, except for floating point instructions
    - You can (and will) write code in C/C++, compile it to MIPS and run it on NachOS
  - Console, Network interface, Timer

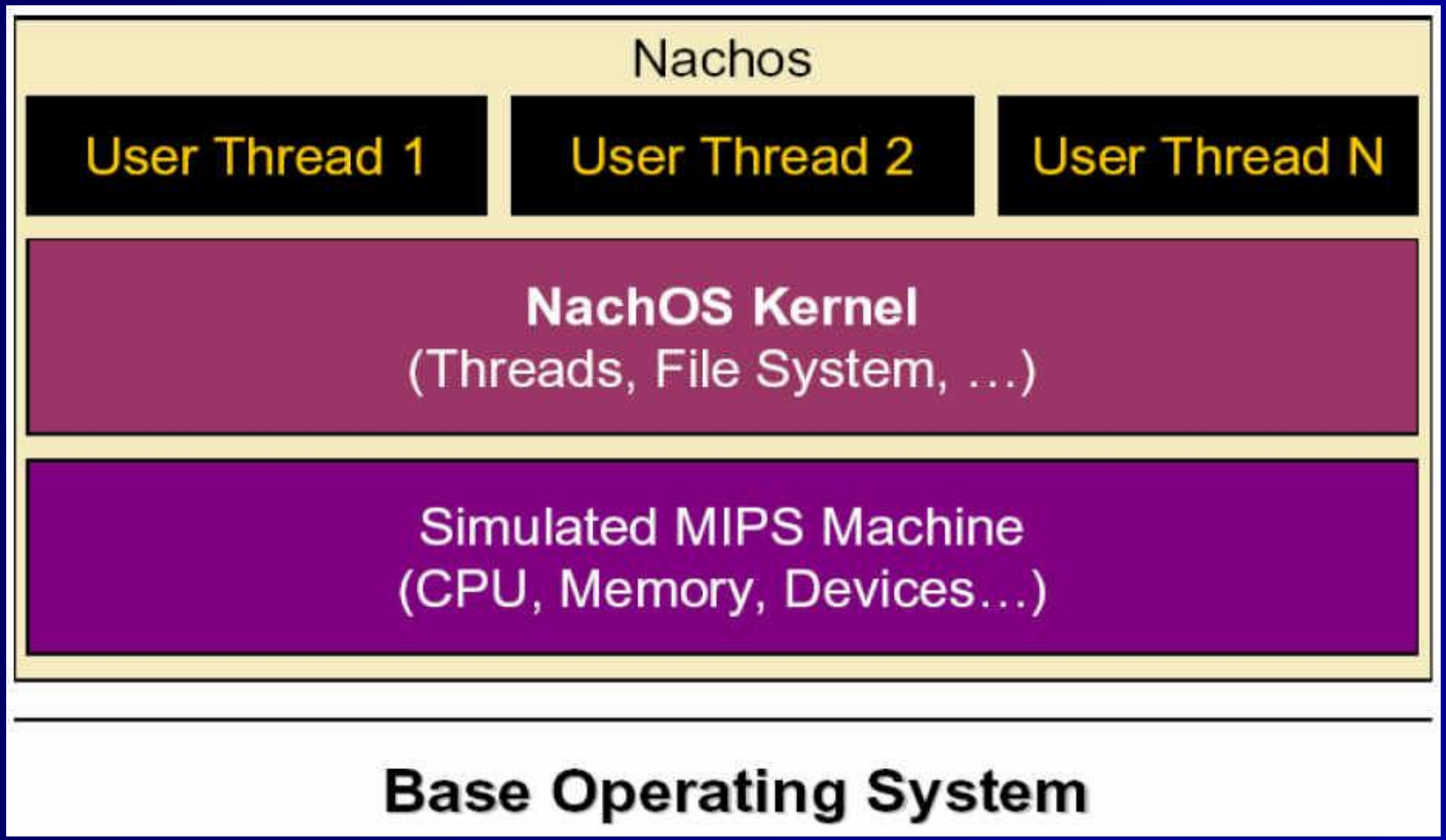
How does it work

- Run as a single UNIX process
- Provide a skeletal OS that supports
  - Threads
  - User-level processes
  - Virtual memory
  - Interrupt-driven I/O devices

How does it work

- Two modes of execution
  - NachOS kernel
    - Executes when
      - NachOS starts up
      - A user-program causes a hardware trap (page fault, syustem call, etc.)
  - MIPS simulator
    - Initialized and started by NachOS kernel

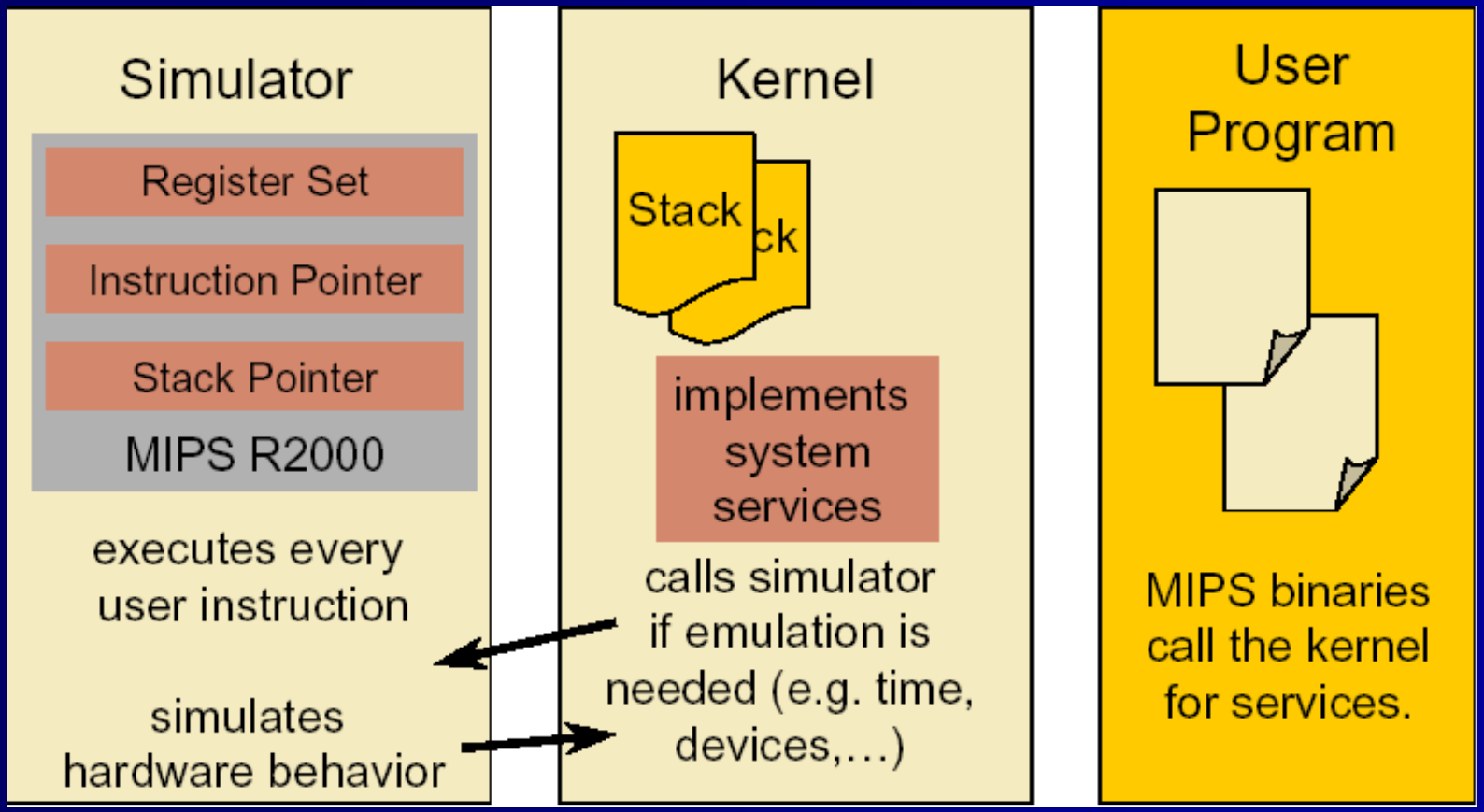
How does it work



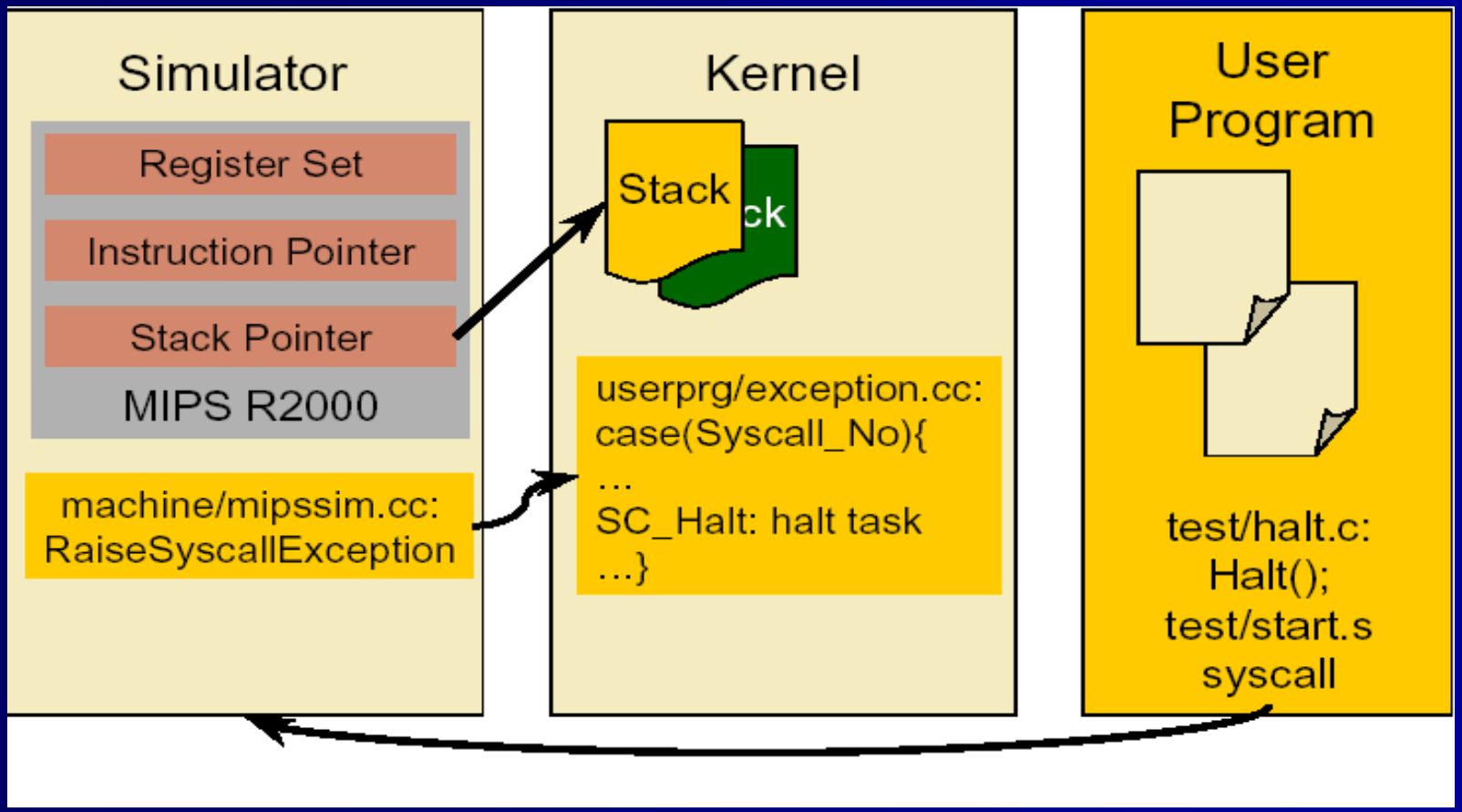
How does it work

- Simulates MIPS architecture on host system (UNIX / Linux / Windows / MacOS X)
- User programs need a cross-compiler (target MIPS)
- NachOS appears as a single threaded process to the host operating system

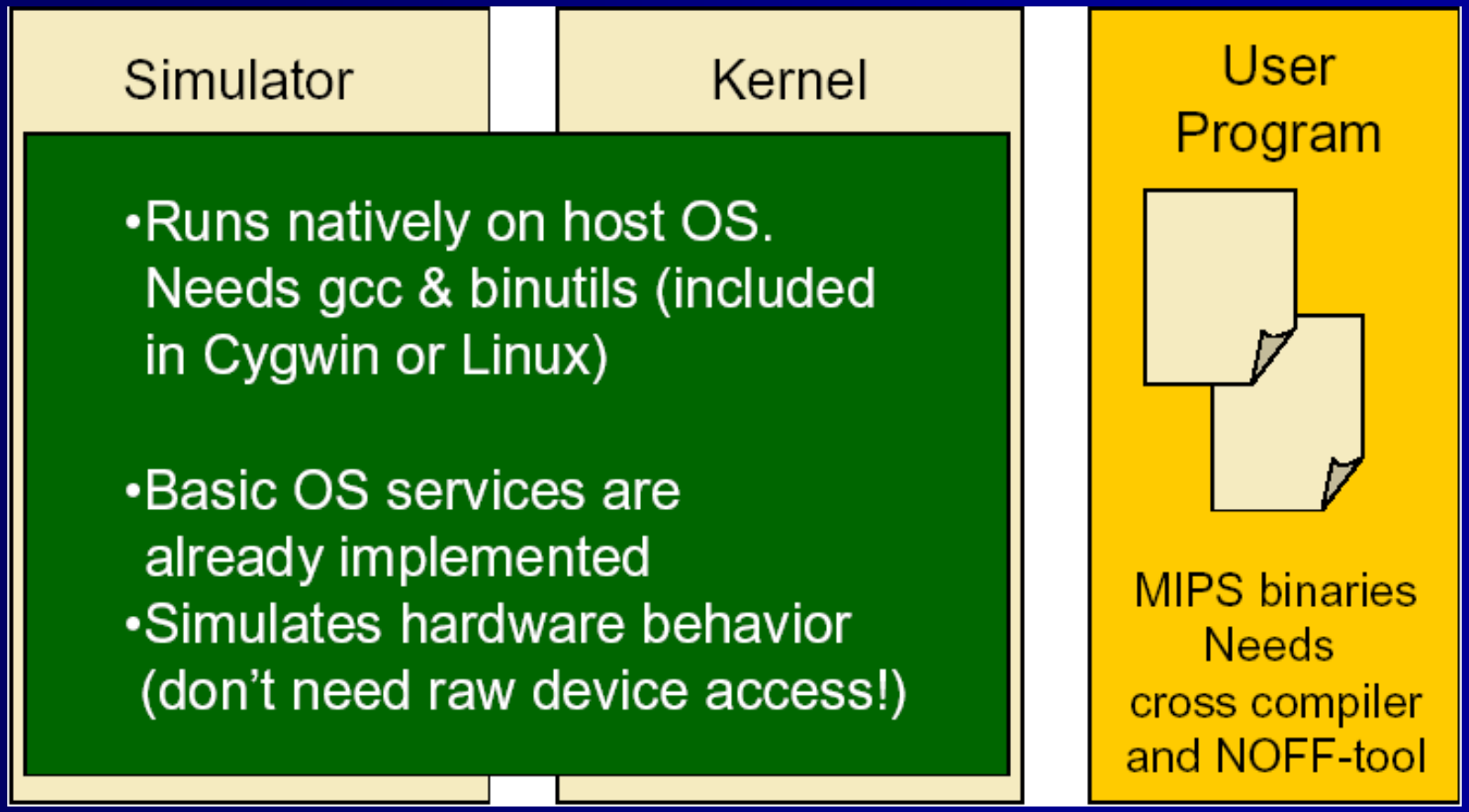
How does it work



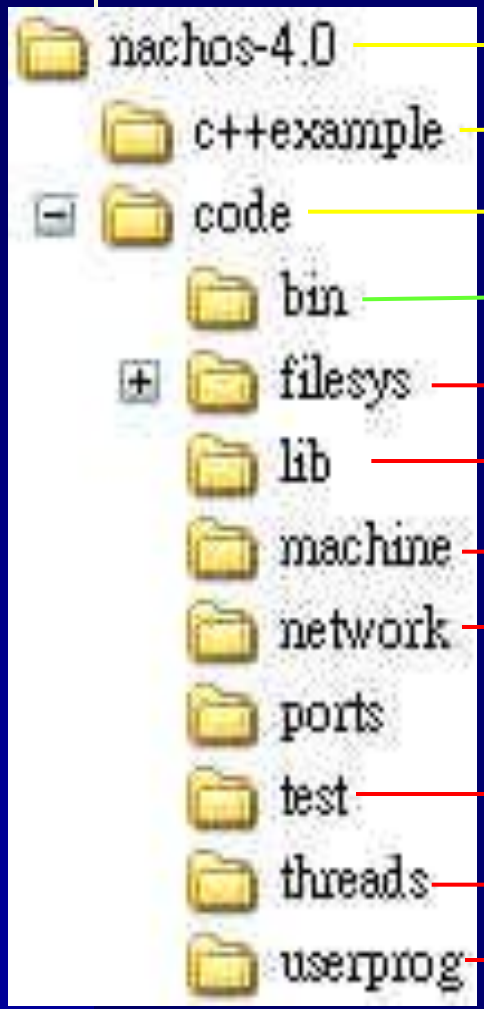
How does it work



How does it work



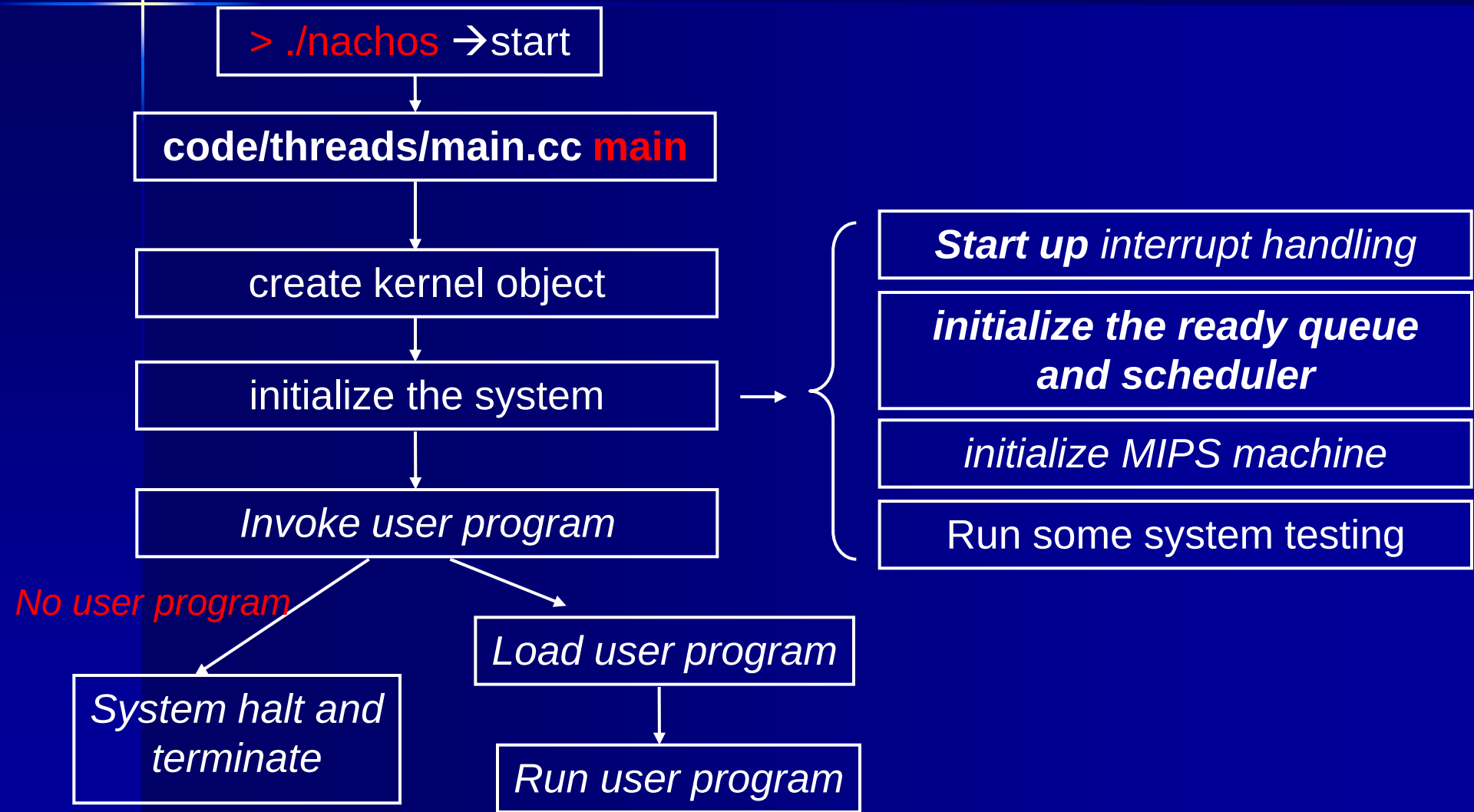
Source Tree



- nachos-4.0 → Root directory
- c++example → C++ introduction to teach how to write C++
- code → Root directory of Nachos's source code
- bin → The tool to convert user programs from MIPS's COFF into NachOS's NOFF (NachOS Object File Format) format
- filesys → File system
- lib → Nachos's library
- machine → MIPS machine
- network → In/Out message queues
- ports
- test → NachOS's sample user programs
- threads → threads
- userprog → User programs' interfaces: system calls, address space, noff format

Source code of NachOS kernel and MIPS simulator

System Start UP



Install NachOS

- Platform: Linux or Linux over VMware
  - RedHat Linux 9.0 (ftp.tku.edu.tw)
- Install steps
  - Get NachOS-4.0
    - wget http://dslab.csie.ncu.edu.tw/OS2006/nachos-4.0.tar.gz
  - Get Cross Compiler
    - wget http://dslab.csie.ncu.edu.tw/OS2006/mips-decstation.linux-xgcc.tgz
  - Move Cross Compiler to /
    - mv ./mips-decstation.linux-xgcc.tgz /
  - Untar Cross Compiler
    - tar zxvf /mips-decstation.linux-xgcc.tgz

Install NachOS

- Untar NachOS-4.0
  - tar zxvf ./nachos-4.0.tar.gz
- Make NachOS-4.0
  - cd ./nachos-4.0/code
  - make
- Test if installation is succeeded
  - cd ./userprog
  - ./nachos -e ../test/test1
  - ./nachos -e ../test/test2
- You should see the following…

The test1 result

```
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
```

The test2 result

```
Total threads number is 1
Thread ../test/test2 is executing.
Print integer:20
Print integer:21
Print integer:22
Print integer:23
Print integer:24
Print integer:25
return value:0
No threads ready or runnable, and no pending interrupts.
Assuming the program completed.
Machine halting!

Ticks: total 200, idle 32, system 40, user 128
Disk I/O: reads 0, writes 0
```

Operate NachOS

- NachOS command help
  - ./nachos -h
- Debugging mode
  - ./nachos -s
- Execute files on NachOS
  - ./nachos -e ../test/test1

Recompile Modified NachOS Code

- cd nachos-4.0/code
- make clean (optional)
- make
  - If you want to fully re-compile the source code, “make clean” is required. Or make will only re-compile the modified and related files. (save time)

Trace NachOS

- Read *.h and *.cc to have an overview about the whole system and see how it is implemented
- Documentation (A Road Map Through NachOS)
  - http://www.cs.duke.edu/~narten/110/nachos/main/main.html

Assignment 1

- NachOS 介紹
- 安裝 NachOS，環境，過程
- 把安裝過程抓圖作成說明文件 (word 檔)
- 把所有在 linux 下用到的指令解釋清楚其用法，並舉例說明

Assignment 2

- 到 nachos-4.0/code/userprog 目錄下執行
  - ./nachos –e ../test/test1 –e ../test/test2

```
Total threads number is 2
Thread ../test/test1 is executing.
Thread ../test/test2 is executing.
Print integer:9
Print integer:8
Print integer:7
Print integer:20
Print integer:21
Print integer:22
```

Assignment 2 (cont.)

- 從上述的輸出，可知結果有誤
- Please trace the following files to see why the output results are wrong
  - nachos-4.0/code/userprog/addrspace.h
  - nachos-4.0/code/userprog/addrspace.cc
  - nachos-4.0/code/userprog/userkernel.cc
  - nachos-4.0/code/machine/translate.h
  - Nachos-4.0/code/machine/translate.cc

Assignment 2 (cont.)

- Please modified the Nachos code to run the correct results
- You may need to modify the following functions
  - nachos-4.0/code/userprog/addrspace.cc
    - constructor
    - destructor
    - initRegister
    - Load

Assignment 3

- Implement system call “Sleep”
- Implement Shortest-Job-First scheduling

System call “Sleep”

- 請實作 Sleep(int x) 這個 system call, 把呼叫這個 system call 的 thread block 住, 並且在 x 次的 timer interrupts 以後才又回到 READY 的狀態。

System call “Sleep” (cont.)

- 修改 exception.cc, syscall.h, start.s
- 呼叫 alarm.cc 的 WaitUntil(int x) 來處理 Sleep(int x) 這個 system call

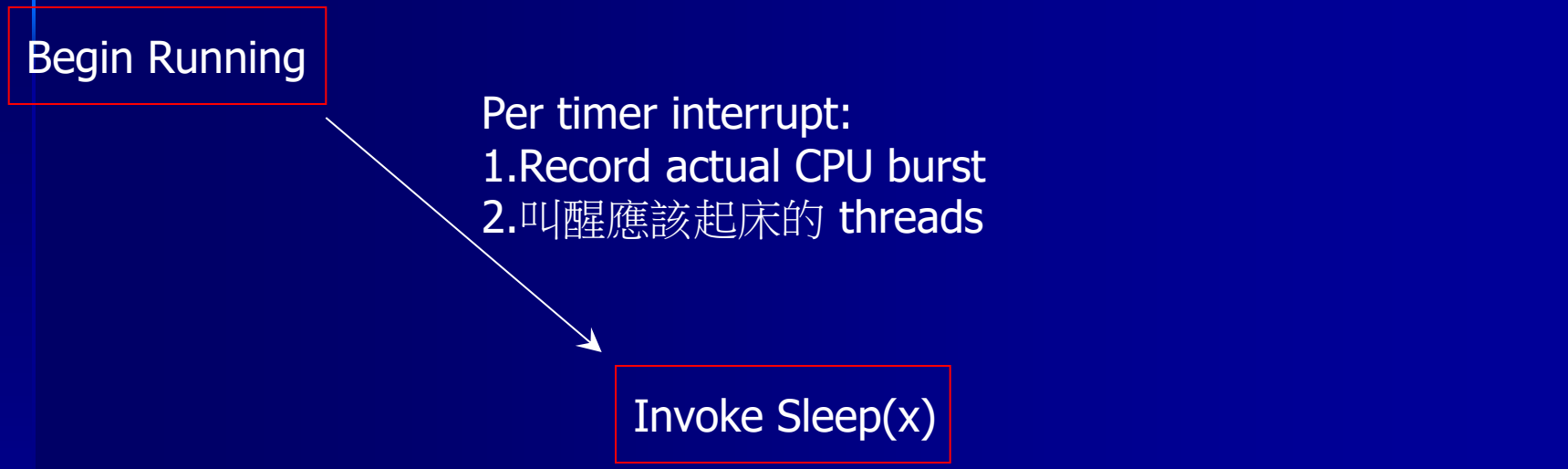
SJF Scheduling

- NachOS 內定的 scheduling algorithm 是 Round-Robin，請設計 non-preemptive SJF Scheduling
- 請自行設計一組或多組 test case 來證明你的 project 是對的，針對 RR 或是 SJF 都要能運作，放到 NachOS 下面去執行的 thread，個數至少 3 個

SJF Scheduling (cont.)

- 由於是 non-preemptive，所以當 timer interrupts 時不用將 thread yield
  - 什麼時候有可能換 thread 執行？
- 需要記錄每個 thread 實際 CPU burst 長度 (timer interrupts 數目), 與預測將來 CPU burst 長度
  - 在哪裡可以得到這個資訊？
  - 何時該進行預測？

SJF Scheduling (cont.)



Demo

- Accompany with your report about what did you done to the NachOS source code (why and how), and what did you get when tracing the NachOS source code.
- 把寫好的source code與報告(word檔)壓縮成”學號.tar.gz”，email to: cyht@dslab.csie.ncu.edu.tw

Demo (cont.)

- 郵件主旨請定為
  - NachOS project report NUM，學號：姓名
- 助教收到 email 後會回信表示收到
  - 如果一週內沒收到回信，請再寄一次