Nachos Project Assignment 2
System Call & CPU scheduling

EE 5173 Operating System
TA: Hsueh-Yi, Chen. Advisor: Farn, Wang.
2023/10/19

Project 2 – part1

System Call – sleep()

System Call – sleep()

Goal:

- Implement a system call.
- Understand how system call work.

System Call – sleep()

What is sleep ?

- The function is used to sleep a thread for a specified amount of time.
- Windows API:
  - sleep(2*1000) # 2 second
- In Unix or POSIX system calls:
  - sleep(2) # 2 second
- Purpose: Slow down your program and can yield other threads to execute

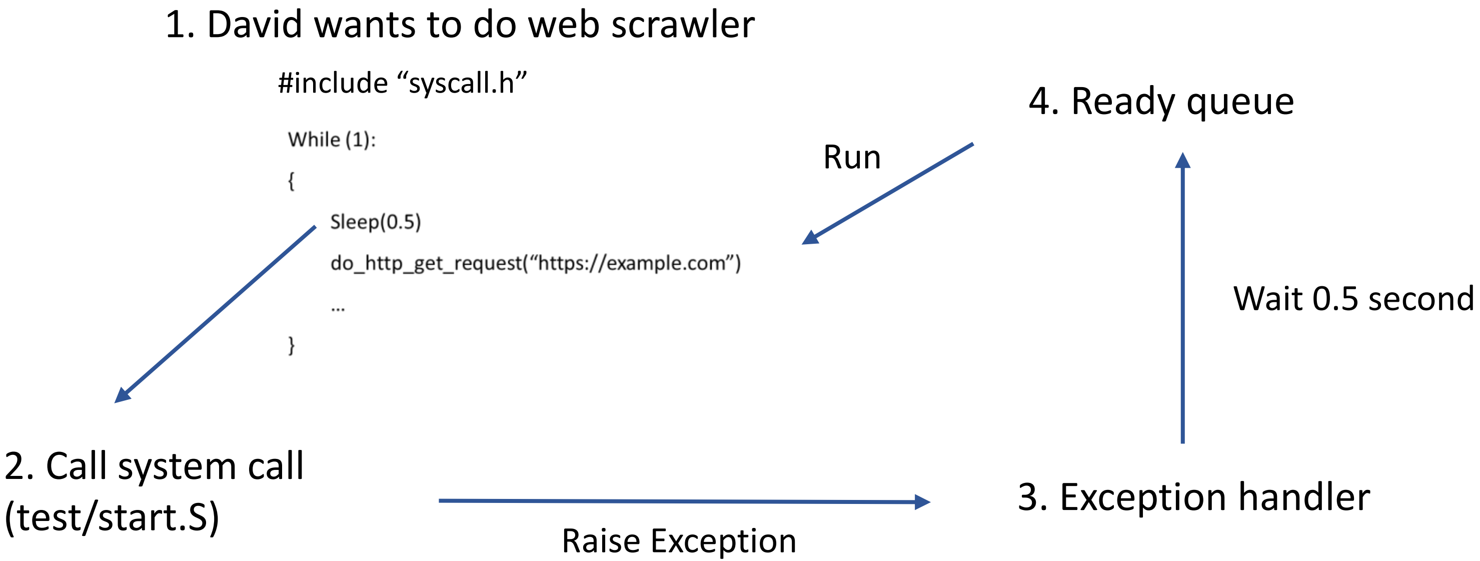
https://en.wikipedia.org/wiki/Sleep_(system_call)

System Call – sleep()

EX: Web scrawler

```
While (1):
{
        Sleep(0.5)
        do_http_get_request(“https://example.com”)
        …
}
```

Sleep syscall execution steps



1. David wants to do web scrawler

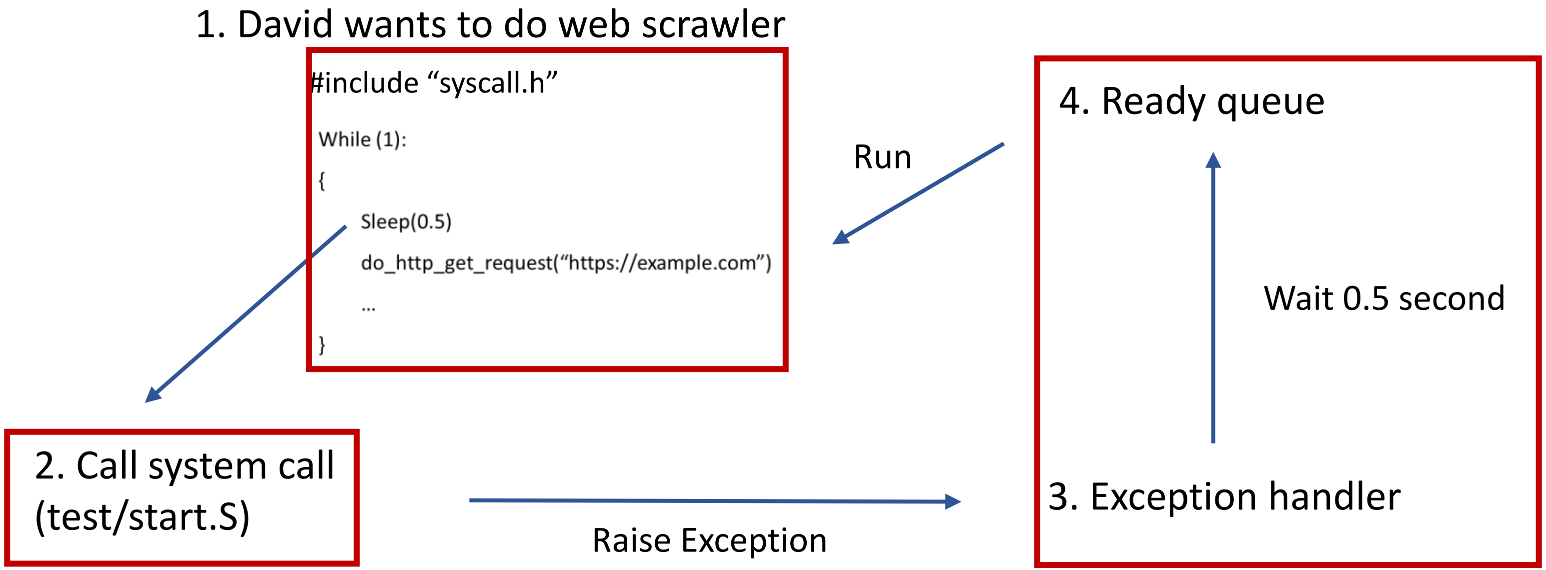
```
#include “syscall.h”

While (1):
{
    Sleep(0.5)
    do_http_get_request(“https://example.com”)
    …
}
```

2. Call system call (test/start.S)

3. Exception handler

4. Ready queue

Sleep syscall execution steps



1. David wants to do web scrawler

```
#include “syscall.h”

While (1):

{

    Sleep(0.5)

    do_http_get_request(“https://example.com”)

    …

}
```

2. Call system call (test/start.S)

Raise Exception

3. Exception handler

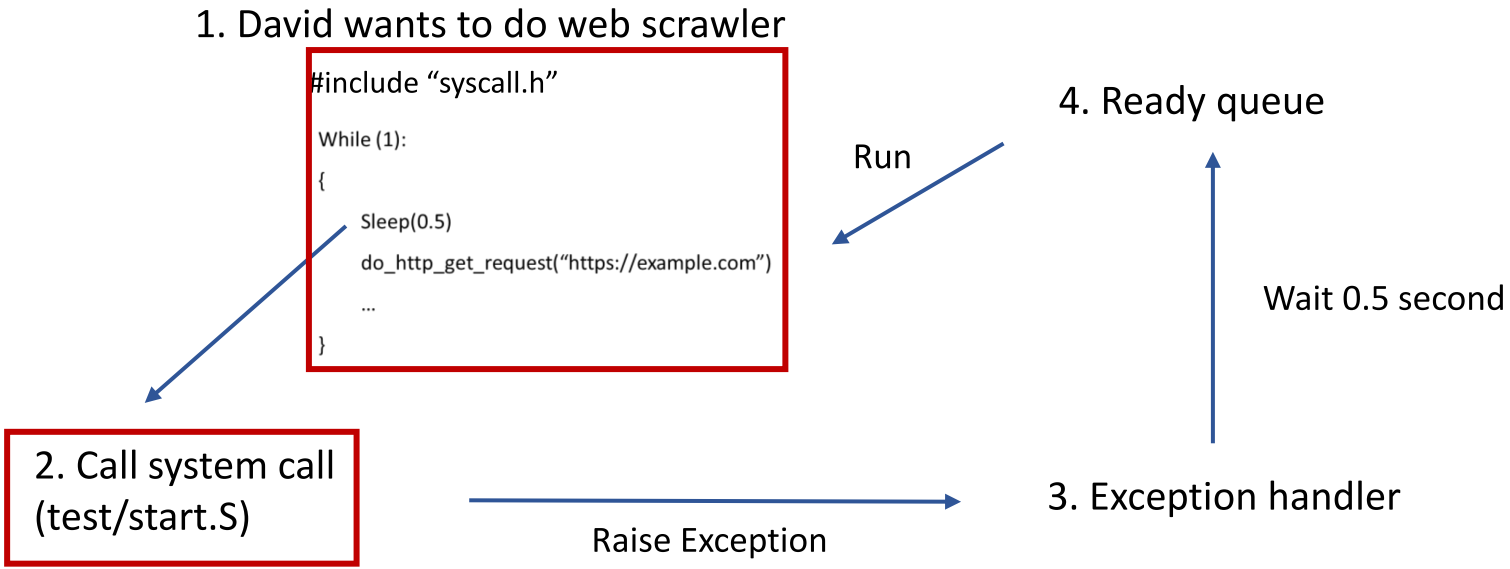
Wait 0.5 second

4. Ready queue

Run

These are what we want to implement.

Sleep syscall execution steps



1. David wants to do web scrawler

```
#include “syscall.h”

While (1):

{

        Sleep(0.5)

        do_http_get_request(“https://example.com”)

        …

}
```

2. Call system call (test/start.S)

Raise Exception

3. Exception handler

Wait 0.5 second

4. Ready queue

Run

System Call – sleep()

- code/userprog/syscall.h

  - System call prototype and number of Sleep.

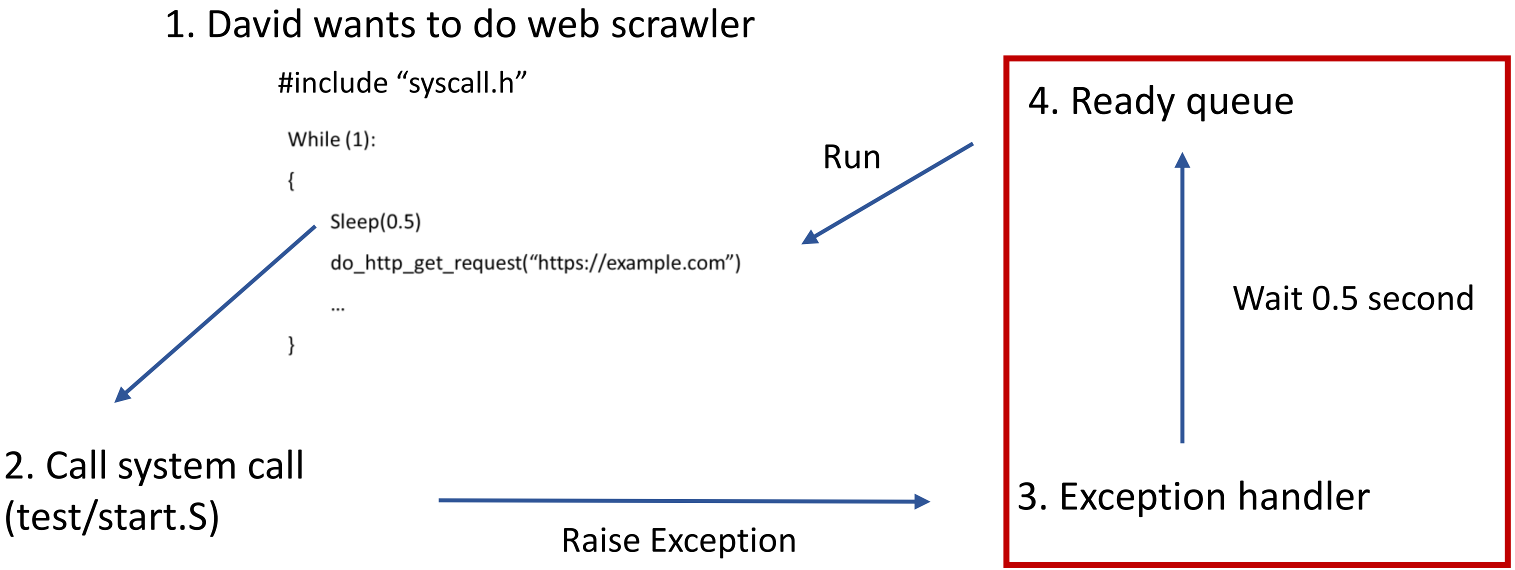
- code/test/start.S

  - Some assembly code help you call system call.

- code/test/test.c

  - your test program

Sleep syscall execution steps



1. David wants to do web scrawler

```
#include "syscall.h"

While (1):

{

    Sleep(0.5)

    do_http_get_request("https://example.com")

    ...

}
```

2. Call system call (test/start.S)

Raise Exception

3. Exception handler

Wait 0.5 second

4. Ready queue

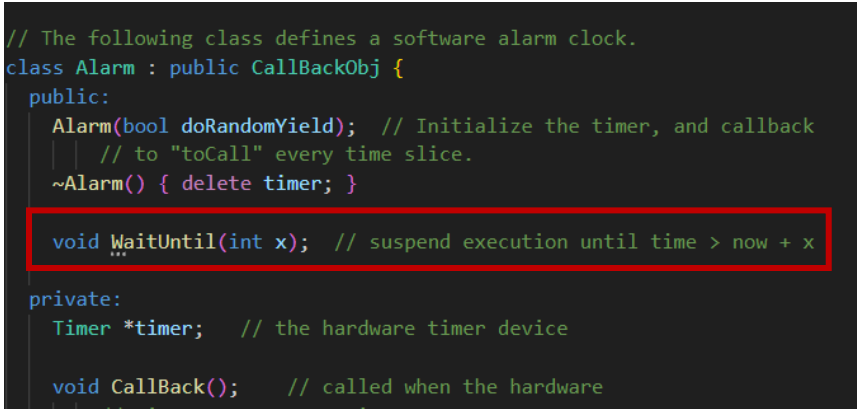
Run

System Call – sleep()

- code/userprog/exception.cc

  - Add new case to handle system call in **ExceptionHandler.**

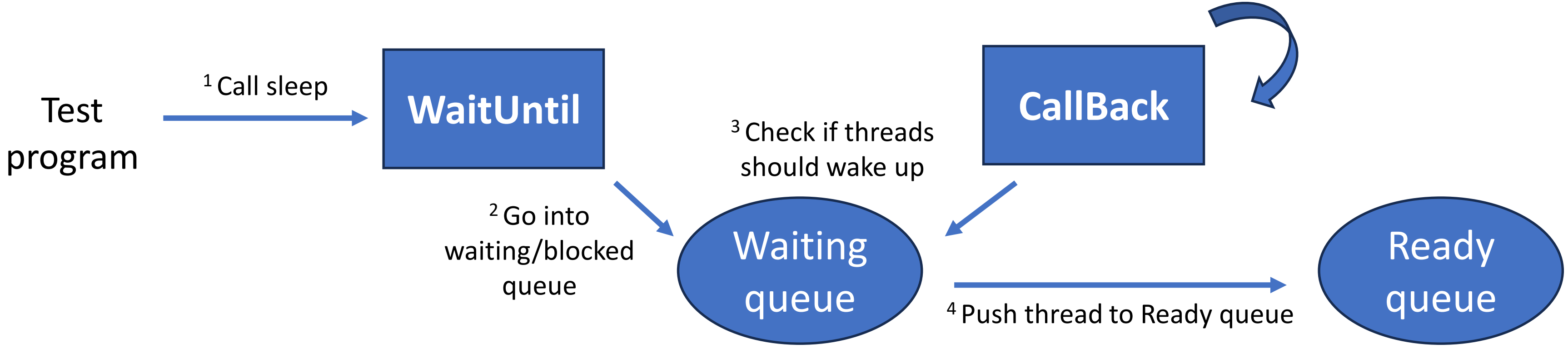
  - Must use ****kernel->alarm->WaitUntil()**



```
// The following class defines a software alarm clock.
class Alarm : public CallBackObj {
  public:
    Alarm(bool doRandomYield);  // Initialize the timer, and callback
        // to "toCall" every time slice.
    ~Alarm() { delete timer; }

    void WaitUntil(int x);  // suspend execution until time > now + x

  private:
    Timer *timer;   // the hardware timer device

    void CallBack();    // called when the hardware
```

System Call – sleep()

- File may modify: /threads/alarm.h, /threads/alarm.cc, /threads/scheduler.cc, /threads/scheduler.cc
- **kernel->alarm->WaitUntil() :** be called when a thread is going to sleep.
- **kernel->alarm->CallBack() :** be called periodically check if threads should wake up (once every one TimerTicks=100)


Test program
[1] Call sleep
WaitUntil
[2] Go into waiting/blocked queue
Waiting queue
[3] Check if threads should wake up
CallBack
[4] Push thread to Ready queue
Ready queue

Some tips

- We should have a class to manage blocked threads and have a list (#include <list>, or sorted_list in lib/list.h) to save the sleep threads
- kernel->stats->totalTicks or other methods (machine/stats.h)

Calling WaitUntil ()

- Push thread and some information to waiting queue
- Push thread to sleep. (threads/thread.h)

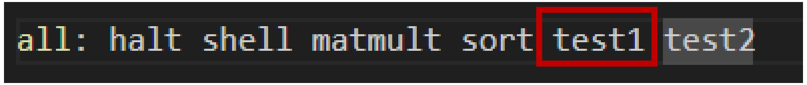
Some tips

Calling CallBack()

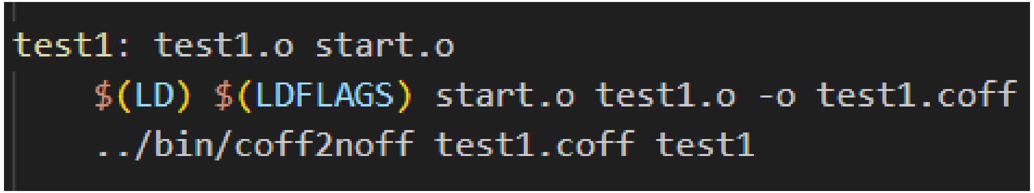
- We should check if there are threads ready to wake up
- Push thread to ready queue (threads/scheduler.h)

How to run test program

- Create a C program in test folder
- Modify Makefile like test1 and test2
- Default “make ” -> make all
- Compile specific file -> run “make test1”



```
all: halt shell matmult sort test1 test2
```



```
test1: test1.o start.o
	$(LD) $(LDFLAGS) start.o test1.o -o test1.coff
	../bin/coff2noff test1.coff test1
```

Project 2 – part2

CPU scheduling

CPU scheduling

Goal:

- Implement some CPU scheduling algorithm.
- Understand how CPU scheduling work.

CPU scheduling

What is the purpose of CPU scheduling?

- Make the system more efficient and quicker when multiprogramming

What is the benefits of CPU scheduling?

- Minimize response time for user
- Minimize the time between submission and finish (turnaround time)
- Minimize total waiting time in ready queue

CPU scheduling

- Choose at least **ONE** of the following to implement:
  - First-Come-First-Service(FCFS)
  - Shortest-Job-First(SJF)
  - Priority
  - Otherwise
- The extra implementation will be considered as **BONUS**.

CPU scheduling

- You can design your test code:
    - You can find Class::SelfTest() in many classes.
    - Implement some test code, and call it in SelfTest()
- Design test case to proof your result put the screenshot in your report.
- Design the nachos interface to switch different scheduling algorithm if you implement more than one. Ex: ./nachos -scheduler FCFS

Some files that might useful

- To change the program interface:
  - threads/main.cc
- To make your own SelfTest() function:
  - threads/thread.h
  - threads/thread.cc
- To call your test code in ThreadedKernel:
  - threads/kernel.cc

Some files that might useful

- Recall Part 1.
  - threads/alarm.h
  - threads/alarm.cc
- Where are the schedulers?
  - threads/scheduler.h
  - threads/scheduler.cc
- Useful data structure
  - E.g. lib/list.h for SortedList.

Questions

1. Explain the details of function call path from Machine::Run to Alarm::CallBack()

Report

- What is your plan? (10%)
- Explain the details of code snippet you added or modified. (40%)
- Experiment result and some discussion, observation (30%)
- What problem you face and tackle it? (10%)
- Questions (10%)

*2 parts in one pdf

* If your code are more different than reference, more score

Policy

- Please save as [Student ID]_project2.pdf
  - E.g. f10921a18_project2.pdf
- Upload to NTU cool. **DDL: 2023/11/16**
- Penalty: decrease 5% per day
- No plagiarism

TAs

- 吳吉加 r12921093@ntu.edu.tw
- 楊冠彥 r11921091@ntu.edu.tw
- 陳學義 f10921a18@ntu.edu.tw

Thanks for your attention !