Introduction to Thread System in Nachos

Linpeng Tang

ACM Honored Class
Shanghai Jiao Tong University

Copyright Announcement

These slides were originally written by Tianqi Chen, and I add some new stuff.

Overview of Tasks

- Build up your thread system.
- Support multi-process nachos.
- Virtual memory support.
- Filesystem design.

Keywords

- Code Reading.
- Design Pattern.
- Multi-Thread Programming.

What is Nachos

An operation system based on a simulate machine

- A simulated MIPS CPU(like SPIM).
- Console and simulated filesystem.
- Thread system based on java-thread.
- Do read the code to find out more.

Code Overview

nachos.machine	Simulation codes, modification not allowed.
nachos.ag	Auto Grader, hope we can improve it better.
nachos.security	Privilege settings, you can skip it in code reading.
nachos.threads	Thread system, start your first phase here .
nachos.userprog	User Process support.
nachos.vm	Virtual Memory support.
nachos.filesys	Filesystem support.
nachos.network	Network support.

Task of Phase 1

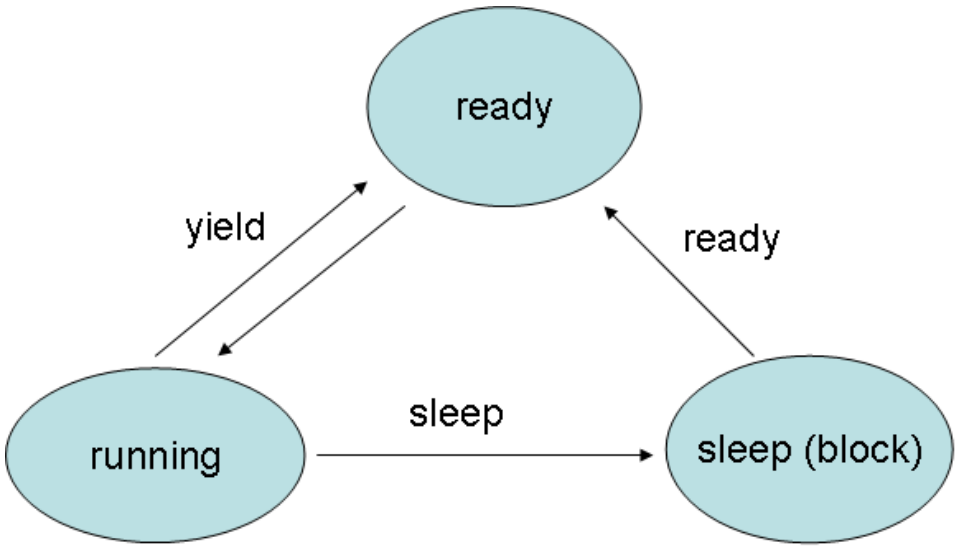
- Build basic tools you need in the next phases.
- Implement different scheduling strategies.
- Use multi-thread programming to solve a problem.

KThread

KThread is the basic thread of nachos

- Only one KThread running in a time.
- **scheduler** decides next thread to run.

Key Functions of KThread



Question: Who calls these functions.

Code Reading: currentThread and this

This is the often confusing point.

- CurrentThread is the thread executing the code.
- this pointer points to the object of the executed function.
- $this \neq currentThread$!

Basic Utilities for Multi-Thread

- Lock
- Semaphore
- Condition
- ...

Scheduler

- RoundRobin Scheduler
- Priority Scheduler
- Lottery Scheduler

Priority Scheduler and Lottery Scheduler shares many things in common, design them together.

ThreadQueue

- Determine the scheduling strategy.
- Not only used in readyQueue !
- Options to transfer priority.

Priority Inversion

Scenario:

- Thread A with priority 1 get Lock A.
- Thread B with priority 3 try to get Lock A, but A already hold the lock, so B have to wait.
- Thread C with priority 2 start to run.

Thread A have no chance to run!

Solution

Transfer priority of A to B temporally.

KThread.Join

Semantics

- If this thread not finished, current thread will sleep (be blocked) until it is finished

Implementation

- Use a ThreadQueue
- No busy waiting

Lock

Method

- acquire()
- release()

Implementation

- Use a ThreadQueue also

Semaphore

Method

- P(), up
- V(), down

Implementation

- Similar to Lock
- Add a counter

Condition Variables

Method

- sleep()
- wake()
- wakeAll()

Implementation

- Take care when wake() is executed just after sleep() but before sleep() finishes.
- The assignment is for you.

Alarm

- Maintain a event queue
- Polling style: check whether a task is due every often

Communicator

- an integer-typed channel
- one message received by exactly one listener
- synchronized communication: speaker(listener) will be blocked until it has sent(received) a message

Integrated Task – Boat

Requirement

- represent each person with a thread
- Top-down simulation allowed

You may get started by thinking how to model the two islands and different states of a person(waiting, rowing, etc.).

Suggestions

- Read the code carefully before you start.
- Read design pattern and understand object orient programming.
- Talk with your friends and us in case you get stuck.
- Enjoy it

Thank you.