实验1：体验Nachos下的并发程序设计

李辉

hui@xmu.edu.cn

厦門大學
XIAMEN UNIVERSITY

实验课信息

- 实验课教师
  - 李辉 （hui@xmu.edu.cn），负责海韵实验楼205
  - 沈思淇（siqishen@xmu.edu.cn），负责海韵实验楼302
- 时间地点
  - 3-15(单)周周四 5-8节
  - 海韵实验楼205、302（1-18组在205机房，19-36组在302机房）

实验课信息

- 三次实验报告及文件提交截止时间分别为4月15日、5月13日及6月10日中午12点。
  - 实验报告和代码以组（每组1-4人）为单位打包发给学委，再由学委统一发给老师（请学委同时发给两位实验课老师）。
  - 是否及时提交以截止时间前课代表是否收到为准。如在截止时间前未提交，则该部分实验报告没有分数。
  - 实验报告应包含以下内容：小组成员姓名学号、实验报告分工、实验设计和关键代码实现、实验结果讨论、遇到的问题、收获和总结。

实验课信息

- 三次实验报告及文件提交截止时间分别为4月15日、5月13日及6月10日中午12点
  - 例如第一次提交的文件：

Makefile.common

main.cc

threadtest.cc

dllist.h

dllist.cc

dllist-driver.cc

nachos01.doc

nachos01.doc为实验报告

实验课信息

- 三次实验报告及文件提交截止时间分别为4月15日、5月13日及6月10日中午12点
  - 实验检查（安排在6月10日课上）对每位同学单独进行，老师和助教会对实验报告和代码进行提问，测试你的理解程度。
  - 评分标准：completeness, correctness, programming style, and thoroughness of testing

实验课信息

- 三次实验报告及文件提交截止时间分别为4月15日、5月13日及6月10日中午12点
  - 如平时不是在实验平台上调试程序（如用个人电脑上的环境），请务必在提交前确认程序在实验平台上可运行！

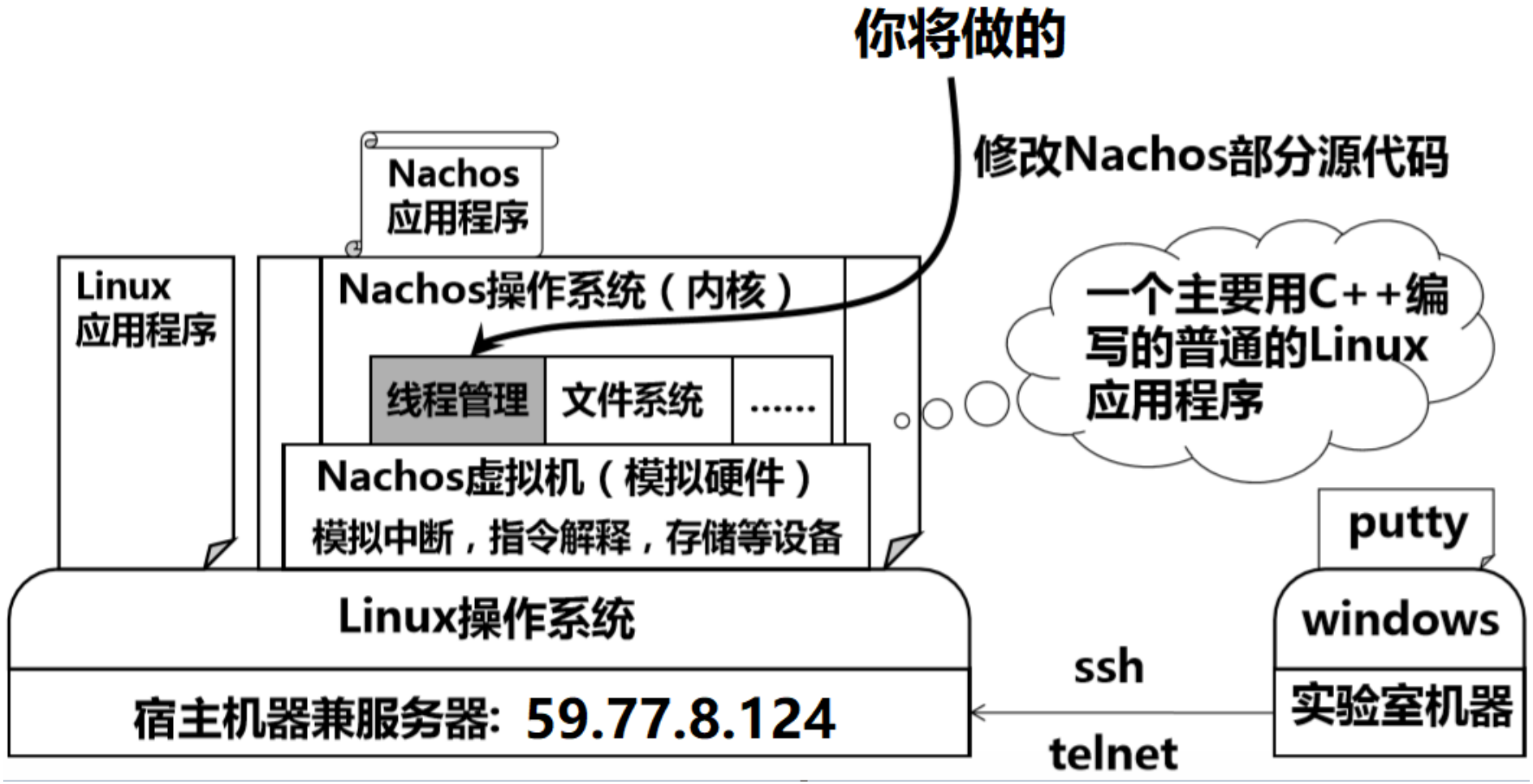
Nachos

- Not Another Completely Heuristic Operating System
- 教学用操作系统
- 原作者：Thomas Anderson （University of California, Berkeley）
- Last Stable Version: 3.4 (1992)
- Last Previous Version: 4 (1996)

Nachos

- 类似于通用虚拟机，便于调试，便于移植；
- 面向对象，用C++的一个子集实现，没有用到面向对象语言的所有特征，如继承性、多态性等；
- 简单而容易扩展；
- 使用并实现了一些新的操作系统概念，如线程、网络和分布式应用；
- 参考：http://homes.cs.washington.edu/~tom/nachos/

Nachos


你将做的
修改Nachos部分源代码
Nachos
应用程序
Linux
应用程序
Nachos操作系统（内核）
线程管理
文件系统
.....
Nachos虚拟机（模拟硬件）
模拟中断，指令解释，存储等设备
一个主要用C++编写的普通的Linux应用程序
putty
windows
实验室机器
Linux操作系统
宿主机器兼服务器: 59.77.8.124
ssh
telnet

Nachos实验1

- 实验目的
  - 熟悉Nachos，初步体验Nachos下的并发程序设计
- 实验内容（具体见《Nachos实验1任务书》和英文文档3.1节）
  - 安装编译Nachos，并阅读源代码
  - 实现双向有序链表（50%）
  - 体验Nachos线程系统（50%）

建议步骤1

- 安装并编译Nachos
  - 首先，确保已会使用Linux的基本命令
    - Linux 常用命令学习 | 菜鸟教程
  - 参考《第一次实验课前准备工作》安装编译Nachos
  - 观察输出结果，思考程序做了什么

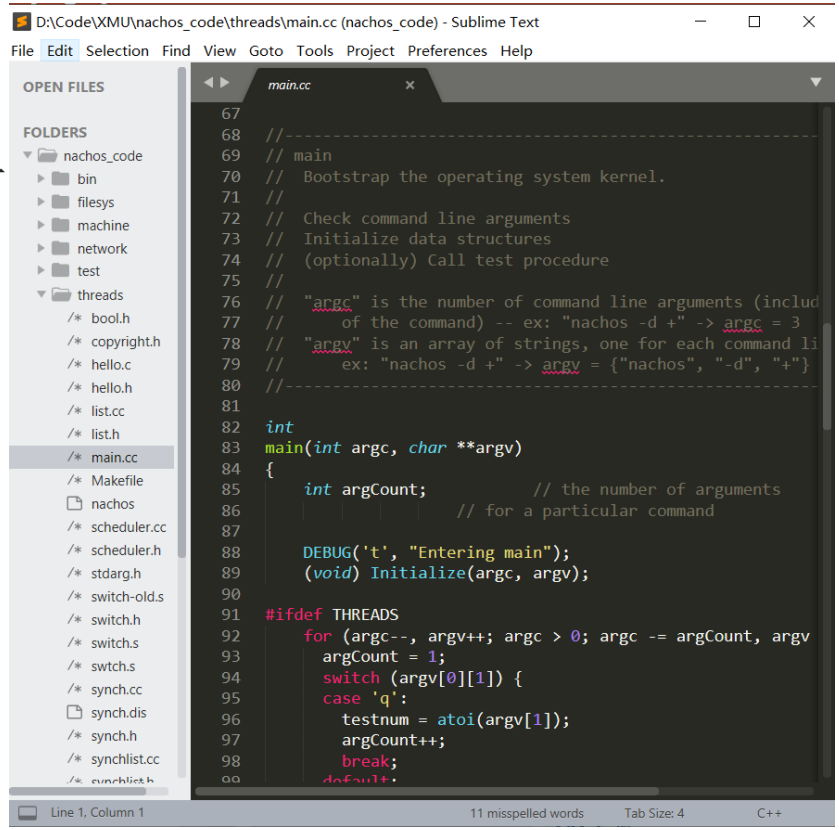
```
[lihui@mcore threads]$ ./nachos
*** thread 0 looped 0 times
*** thread 1 looped 0 times
*** thread 0 looped 1 times
*** thread 1 looped 1 times
*** thread 0 looped 2 times
*** thread 1 looped 2 times
*** thread 0 looped 3 times
*** thread 1 looped 3 times
*** thread 0 looped 4 times
*** thread 1 looped 4 times
No threads ready or runnable, and no pending interrupts.
Assuming the program completed.
Machine halting!

Ticks: total 130, idle 0, system 130, user 0
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

Cleaning up...
[lihui@mcore threads]$
```

建议步骤2

- 阅读Nachos英文文档
  - 我需要修改/新增Nachos哪一部分的代码？我不该随意改动哪些代码？（1.7节）
  - code文件夹下的子目录都放了啥？输入“make depend”、“ make”、和“./nachos” 都发生了什么？（2.1节）
  - 我如何Debug Nachos代码？（2.2节）
  - 如何在Nachos进行context switch？（2.3节）

建议步骤3

- 进一步了解Nachos源代码
  - 很多工具可用于阅读和编写代码
    - 如用Sublime Text 3配合SFTP插件可调用远程服务器环境编写代码
    - Sublime + SFTP使用
    - SFTP离线安装



建议步骤3

- 进一步了解Nachos源代码
  - 阅读nachos-3.4/code/目录下的Makefile, Makefile.dep, Makefile.common，以及nachos-3.4/code/threads/Makefile，初步了解Makefile的构成和相互关系（需掌握make用法，参考《跟我一起写makefile》及其他网络资料）
  - 阅读nachos-3.4/code/threads/main.cc了解Nachos如何运行

建议步骤3

- 进一步了解Nachos源代码
  - 阅读nachos-3.4/code/threads/system.cc的Initialize函数中的与 debug相关的部分及nachos-3.4/code/threads/utility.cc，了解DEBUG的实现与使用；（可选）
  - 阅读nachos-3.4/code/threads/threadtest.cc，理解Nachos中线程的概念（用户级还是内核级线程？）及其运行方式。

建议步骤4

- 初步尝试修改Nachos源代码
  - 写一个最简单的程序，比如包含一个可以输出“Hello Nachos”的Hello函数，函数实现写在 hello.c里，函数声明写在hello.h里；
  - 将hello的代码文件拷贝到nachos3.4/code/threads/中，相应修改nachos3.4/code/Makefile.common中的 THREAD_H、THREAD_C、THREAD_O 所在行的相关内容；

建议步骤4

- 初步尝试修改Nachos源代码
  - 修改nachos-3.4/code/threads/main.cc使之包含hello.h头文件，并在main函数中调用ThreadTest 函数之前添加对Hello函数的调用；（ ThreadTest 是啥可阅读英文文档3.1.2节）
  - 在nachos-3.4/code/threads/中依次执行“make depend”和“ make”。

建议步骤4

- 初步尝试修改Nachos源代码
  - 在nachos-3.4/code/threads/中运行“./nachos”；

```
user@virtual-ubuntu-server:~/nachos-3.4/code/threads$ ./nachos
Hello Nachos
*** thread 0 looped 0 times
*** thread 1 looped 0 times
*** thread 0 looped 1 times
*** thread 1 looped 1 times
*** thread 0 looped 2 times
*** thread 1 looped 2 times
*** thread 0 looped 3 times
*** thread 1 looped 3 times
*** thread 0 looped 4 times
*** thread 1 looped 4 times
No threads ready or runnable, and no pending interrupts.
Assuming the program completed.
Machine halting!

Ticks: total 130, idle 0, system 130, user 0
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

Cleaning up...
```

ThreadTest

建议步骤5

- 用C++实现双向链表
  - 需自行编写三个文件dllist.h、dllist.cc和dllist-driver.cc；
  - 给定头文件dllist.h内容提供在Nachos英文文档的3.1.1 章节；
  - 仔细阅读Nachos英文文档的3.1.1章节并按照其描述完成本实验步骤；
  - 学习gcc与g++编译器的使用方法（可自行查阅网络资料）

建议步骤6

- 修改Nachos源代码体验并发程序问题
  - 修改nachos-3.4/code/threads/threadtest.cc和nachos3.4/code/threads/main.cc，重新编译threads子系统，使得链表操作可以在Nachos中运行并实现用两个线程并发操作链表。

建议步骤6

- 修改Nachos源代码体验并发程序问题
  - 自行设计可能使并发链表操作发生错误的线程执行顺序，考虑3-5 种不同的顺序为宜；
  - 实现以上执行顺序：修改nachos-3.4/code/threads/threadtest.cc，在“适当”的位置（使得可以观察到并发程序问题）插入 currentThread->Yield()调用以强制线程切换（注意相应文件中应包含对外变量currentThread的声明并include头文件thread.h）， 重新编译threads子系统并运行观察出现的问题。

注意事项

- Unix命令、make与makefile的用法、c++编程，除课上提供的参考资料外，可自行网上搜索相关资料学习；
- 阅读Nachos源代码时注意不要忽略注释里的内容；
- 可通过Nachos英文文档了解很多细节信息；
- 小组内部应互帮互助，积极（通过微信、QQ群）交流讨论；
- 利用GitHub等平台协同工作，同步代码。

第一次实验课后可能遇到的一些问题

- 我在XX环境下编译的时候遇到编译错误（可能性1）
  - 首先用命令“gcc -v”和“g++ -v”查看编译器版本
    - 实验平台：4.1.2
  - 检查是否在上别的课程时，在实验平台自己的用户目录下装了额外的编译器。如果是的话将gcc和g++的链接改为系统默认的/usr/bin/gcc和/usr/bin/g++。
    - 怎么改？有可能是在用户目录下的.bashrc文件里将gcc和g++的路径指向了自己安装的编译器目录

第一次实验课后可能遇到的一些问题

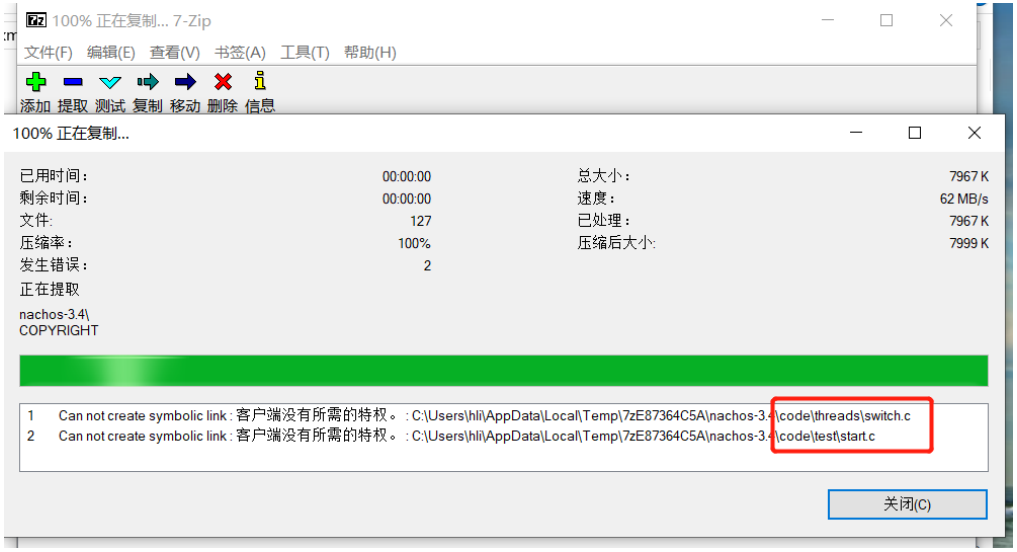
- 我在XX环境下编译的时候遇到编译错误 （可能性2）
  - 修改了一写代码后，觉得没有改错但是编译出错了。可以先进入threads文件夹，用“rm *.o”命令删除threads目录下的所有目标文件，再重新编译。

第一次实验课后可能遇到的一些问题

- 我在XX环境下编译的时候遇到编译错误（可能性3）
  - 是否是在实验平台上用命令解压的“nachos-linux64.tar.gz”压缩包？
  - 如果用一些不完善的解压软件直接在Windows下解压然后通过工具上传到实验平台，解压过程不会报错，但是实际上解压的结果是错误的！

第一次实验课后可能遇到的一些问题

- 我在XX环境下编译的时候遇到编译错误（可能性3）
  - 如果是在Windows下用比较完善的解压软件如7-Zip直接解压，会提示你switch.c和start.c没有权限，因为其实这两个文件是Linux上的“链接文件”。



第一次实验课后可能遇到的一些问题

- 我在XX环境下编译的时候遇到编译错误（可能性3）
  - 进入code/threads和code/test，用命令“ls -il”检查下switch.c和start.c是否正确的解压为“链接”文件了

正确的解压结果

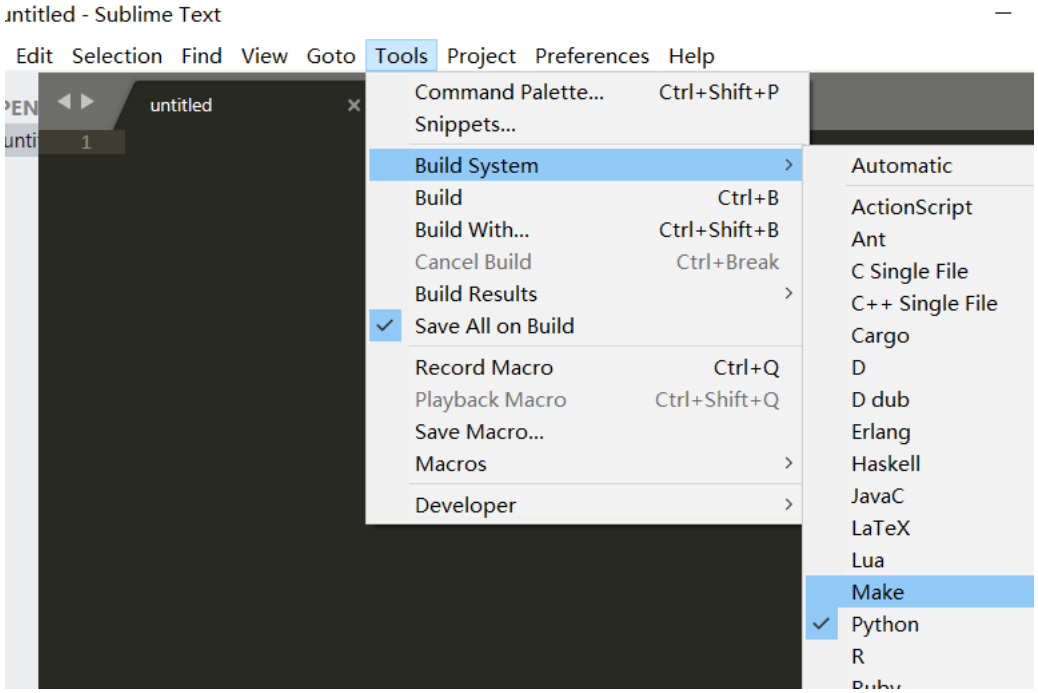
```
[lihui@mcore threads]$ ls -il
总计 256
2231911 -rw-r----- 1 lihui lihui    91 1999-10-05 bool.h
2231921 -rw------- 1 lihui lihui  1229 1993-10-04 copyright.h
2231916 -rw------- 1 lihui lihui  7143 1993-10-04 list.cc
2231920 -rw------- 1 lihui lihui  2298 1993-10-04 list.h
2231899 -rw------- 1 lihui lihui  5479 2001-10-02 main.cc
2231903 -rw------- 1 lihui lihui 12121 2005-01-15 Makefile
2231901 -rw------- 1 lihui lihui  5115 1993-10-04 scheduler.cc
2231918 -rw------- 1 lihui lihui  1215 1993-10-04 scheduler.h
2231904 -rw-r----- 1 lihui lihui  5794 1999-10-05 stdarg.h
2231905 lrwxrwxrwx 1 lihui lihui     8 04-18 10:17 switch.c -> switch.s
2231906 -rw------- 1 lihui lihui  3806 2000-10-07 switch.h
```

错误的解压结果

```
[lihui@mcore threads]$ ls -il
总计 664
32047521 -rw-r--r-- 1 lihui lihui    91 04-18 10:06 bool.h
32047539 -rw-r--r-- 1 lihui lihui  1229 04-18 10:06 copyright.h
32047507 -rw-r--r-- 1 lihui lihui  3395 04-18 10:06 dllist.cc
32047540 -rw-r--r-- 1 lihui lihui   661 04-18 10:06 dllist-driver.cc
32047532 -rw-rw-r-- 1 lihui lihui  5624 04-18 10:06 dllist-driver.o
32047538 -rw-r--r-- 1 lihui lihui  1454 04-18 10:06 dllist.h
32047535 -rw-rw-r-- 1 lihui lihui 10808 04-18 10:06 dllist.o
32047522 -rw-rw-r-- 1 lihui lihui 17204 04-18 10:06 interrupt.o
32047529 -rw-r--r-- 1 lihui lihui  7143 04-18 10:06 list.cc
32047537 -rw-r--r-- 1 lihui lihui  2298 04-18 10:06 list.h
32047497 -rw-rw-r-- 1 lihui lihui  7792 04-18 10:06 list.o
32047498 -rw-r--r-- 1 lihui lihui  5792 04-18 10:06 main.cc
32047531 -rw-rw-r-- 1 lihui lihui  5204 04-18 10:06 main.o
32047508 -rw-r--r-- 1 lihui lihui 15285 04-18 10:06 Makefile
32047515 -rwxrwxr-x 1 lihui lihui 111117 04-18 10:06 nachos
32047502 -rw-r--r-- 1 lihui lihui  5115 04-18 10:06 scheduler.cc
32047533 -rw-r--r-- 1 lihui lihui  1215 04-18 10:06 scheduler.h
32047505 -rw-rw-r-- 1 lihui lihui 10000 04-18 10:06 scheduler.o
32047524 -rw-rw-r-- 1 lihui lihui  4296 04-18 10:06 stats.o
32047509 -rw-r--r-- 1 lihui lihui  5794 04-18 10:06 stdarg.h
32047511 -rw-r--r-- 1 lihui lihui     8 04-18 10:06 switch.c
32047513 -rw-r--r-- 1 lihui lihui  3806 04-18 10:06 switch.h
32047501 -rw-rw-r-- 1 lihui lihui   700 04-18 10:06 switch.o
```

第一次实验课后可能遇到的一些问题

- 我用命令行调用make命令没有问题，用Sublime的make命令报错
  - Sublime只是个文本编辑器，编译出错的原因一定是配置问题或者代码问题



第一次实验课后可能遇到的一些问题

- 我用命令行调用make命令没有问题，用Sublime的make命令报错
  - Sublime不支持调用远程make，执行make命令的时候是调用本地的make。即使要调用本地make命令，也需要进行配置。如参考：sublime text3设置运行自己的Makefile。
  - 建议本地编程和与实验平台传代码文件用Sublime，编译还是在命令行输入命令进行。

第一次实验课后可能遇到的一些问题

- 我不习惯Sublime，我平时用XXX
  - 你可以用任意你喜欢的编辑器/IDE完成实验，第一节课演示Sublime和提供相应资料是为了提供给不知道用什么工具看代码的同学。
  - 实验检查的时候，你可以用任何你习惯的工具（甚至是命令行）打开代码演示。

开始动手实验

- 大家现在可利用实验课时间完成实验任务
- 有问题可以提问，老师进行解答

谢谢

Hui Li | hui@xmu.edu.cn