Housekeeping (Lecture 2 - 1/13,14/2016)

- Warmup #1 due at 11:45pm on Friday, 1/29/2016
  - if you have code from a previous semester, be very careful and *not copy any code from it*
    - it's best if you just get rid of it
  - get started soon
    - if you are stuck, make sure you come to see me during office hours or send me e-mail
    - feel free to discuss over the class Google Group
- The TAs will introduce warmup #1 to you this Friday
  - please understand that discussion section material are *NOT* substitute for reading the specs and the grading guidelines
    - you are expect to read the entire *spec*
    - you are expect to read the *requirements* the spec refers to
    - you are expect to read the *grading guidelines*
    - it's your responsibility

Housekeeping (Lecture 2 - 1/13,14/2016)

- *Grading guidelines* is the *ONLY* way we will grade and we can only grade on `nunki.usc.edu` in our grading account (which you don't have access to)
  - due to our *fairness* policy
  - it's a very good idea to run your code against the grading guidelines on `nunki.usc.edu`
    - there are some differences between Unix and Linux
  - the *grading guidelines* is *part of the spec*
- If you make a submission
  - read and understand the output of `bsubmit`
  - make sure you follow the *"Verify Your Submission"* procedure

Housekeeping (Lecture 2 - 1/13,14/2016)

- Tentative timeline for warmup #1
  - you should be done with part (A) of the grading guidelines by next Tuesday (one week before the extra credit deadline)
  - you should finish warmup #1 before the extra credit deadline
- If you want a good grade, it's important that you keep up with the lectures
  - if you don't understand anything that's covered in class, come see me or send me e-mail
- You need to learn Unix!
  - our kernel assignments are to implement a Unix system!
  - you don't need to be an Unix expert, you just need to know the basics, e.g., directory listing, creating directory, change directory, copy file, delete file, etc.

Housekeeping (Lecture 2 - 1/13,14/2016)

- You should install Ubuntu 12.04
  - and start using it for warmup #1
- You should start looking for partners for kernel assignments
  - work with your potential partners on warmups 1 and 2
    - again, work at high level and must *not* share code
- This class does not use DEN or the Blackboard system
  - except for lecture videos on DEN
  - everything you need is on the class web site
    - http://merlot.usc.edu/cs402-s16
  - please spend some time getting familiar with the class web site, especially with all the *rules about grading*

Housekeeping (Lecture 4 - 1/20,21/2016)

- Warmup #1 due at 11:45pm on Friday, 1/29/2016
  - if you have code from a previous semester, be very careful and *not copy any code from it*
    - it's best if you just get rid of it
  - according to my tentative timeline, you should be done with part (A) of the grading guidelines by now
    - if you are stuck, make sure you come to see me/TA/CP during office/helpdesk hours or send us e-mail
    - feel free to discuss over the class Google Group
- *Grading guidelines* is the *ONLY* way we will grade and we can only grade on `nunki.usc.edu` in our grading account (which you don't have access to)
  - the *grading guidelines* is *part of the spec*
  - it's a very good idea to run your code against the grading guidelines on `nunki.usc.edu`
    - there are some differences between Unix and Linux

Housekeeping (Lecture 4 - 1/20,21/2016)

- If you make a submission
  - read and understand the output of `bsubmit`
  - make sure you follow the *"Verify Your Submission"* procedure
- You should install Ubuntu 12.04
  - and start using it for warmup #1
    - `valgrind` is a great tool but it only runs on Linux machines
  - if you have a fast enough machine, my favorite way to install Ubuntu 12.04 was to first install *VMware Player 7* for *Windows* (or VirtualBox for Macs)
    - then install Ubuntu 12.04 into it
- If you are looking for kernel partners, please go to the projects web page
- Starting next week, MW classes are 80 minutes long (will go back to 110 minutes on 2/8/2016 for 3 lectures)

Housekeeping (Lecture 5 - 1/25,26/2016)

- Warmup #1 due at 11:45pm this Friday, 1/29/2016
  - if you have code from a previous semester, be very careful and *not copy any code from it*
    - it's best if you just get rid of it
  - if you are confused about any part of warmup #1, you need to come to office/helpdesk hours!
- *Grading guidelines* is the *ONLY* way we will grade and we can only grade on `nunki.usc.edu` in our grading account (which you don't have access to)
  - we will use a different set of *data files* to grade, but we won't change the grading scripts
- If you make a submission
  - make sure you follow the *"Verify Your Submission"* procedure
- Still quite a few students on the waiting list
  - it's probably a good idea to have a backup plan in case you don't get in by this Friday

Housekeeping (Lecture 5 - 1/25,26/2016)

- Do GDB Assignment #1
  - IMPORTANT: draw picture on a piece of paper!
  - first, change "`num_items=64`" in `DoTest()` to "`num_items=3`"

```
make
gdb listtest
(gdb) break DoTest
(gdb) run
(gdb) n                                  <-- do this 5 times, you are now at call to CreateTestList()
(gdb) print list                         <-- does the list look like an empty list?
(gdb) n                                  <-- returned from CreateTestList()
(gdb) print list                         <-- does the list look like a 3-item list?
(gdb) print &(list.anchor)               <-- what's the address of the anchor?
(gdb) print list.anchor                  <-- what's in the anchor?
(gdb) print *(list.anchor.next)
(gdb) print *(list.anchor.next->next)
(gdb) print *(list.anchor.next->next->next)
```

this should be the last list element, does its `next` pointer point to the anchor?

Housekeeping (Lecture 6 - 1/27,28/2016)

- Warmup #1 due at 11:45pm this Friday, 1/29/2016
  - if you have code from a previous semester, be very careful and *not copy any code from it*
    - it's best if you just get rid of it
  - if you are confused about any part of warmup #1, you need to come to office/helpdesk hours!
- *Grading guidelines* is the *ONLY* way we will grade and we can only grade on `nunki.usc.edu` in our grading account (which you don't have access to)
  - we will use a different set of *data files* to grade, but we won't change the grading scripts
- If you make a submission
  - make sure you follow the *"Verify Your Submission"* procedure

Housekeeping (Lecture 6 - 1/27,28/2016)

- Still quite a few students on the waiting list
  - it's probably a good idea to have a backup plan in case you don't get in by this Friday
- There is no roll sheet signing for *lectures* this semester
  - everyone gets 2% extra credit for free
  - the other 2% extra credit is for signing roll sheets in discussion sections
    - only if you sign roll sheets for the discussion section you are *registered*
    - this starts *next week* (i.e., week 4 of the semester)
- You should do GDB Assignment #1
  - IMPORTANT: draw picture on a piece of paper!

Housekeeping (Lecture 7 - 2/1,2/2016)

- Warmup #2 due at 11:45pm on Friday, 2/19/2016
  - if you have code from a previous semester, be very careful and *not copy any code from it*
    - it's best if you just get rid of it
  - start early
- *Grading guidelines* is the *ONLY* way we will grade and we can only grade on `nunki.usc.edu` in our grading account
- If you make a submission
  - make sure you follow the *"Verify Your Submission"* procedure
- You should start looking for partners for kernel assignments
  - if you want to be part of a team, add your information to http://merlot.usc.edu/cs402-s16/projects/groups/
  - work with your potential partners on warmup 2
    - again, work at high level and must *not* share code
  - team forming deadline is 2 days after warmup 2 is due

Housekeeping (Lecture 7 - 2/1,2/2016)

- Recommended timeline for warmup #2
  - don't worry about <Cntrl+C> during the first week
  - make the first procedure of threads *nice and simple*
    - make a bunch of well-defined function calls
    - write *pre-conditions* and *post-conditions* in a comment block right above each of these functions
  - use just *one mutex* to lock and unlock the *entire "token bucket filter" data structures* that's shared by all the threads
  - get the simulation/emulation to work during the first week (and before Tuesday of next week)
  - write small programs to test out ideas
  - the lecture today should cover everything you need except for <Cntrl+C> handling
    - you can add <Cntrl+C> handling code next week
  - by the end of this week, you will know everything you need to complete warmup #2

Housekeeping (Lecture 8 - 2/3,4/2016)

- Warmup #2 due at 11:45pm on Friday, 2/19/2016
  - if you have code from a previous semester, be very careful and *not copy any code from it*
    - it's best if you just get rid of it
  - start early
- *Grading guidelines* is the *ONLY* way we will grade and we can only grade on `nunki.usc.edu` in our grading account
- If you make a submission
  - make sure you follow the *"Verify Your Submission"* procedure
- For MW section, starting next Monday, 3 lectures will be 110 minutes long
  - 2/15/2016 is a holiday
- Office hour this Thursday moved to 2:30-3:30pm

Housekeeping (Lecture 8 - 2/3,4/2016)

- Recommended timeline for warmup #2
  - make the first procedure of threads *nice and simple*
  - use just *one mutex* to lock and unlock the *entire "token bucket filter" data structures* that's shared by all the threads
  - get the simulation/emulation to work before Tuesday of next week
  - add <Cntrl+C> handling code with thread cancellation next week

Housekeeping (Lecture 9 - 2/8,9/2016)

- Warmup #2 due at 11:45pm on Friday, 2/19/2016
  - if you have code from a previous semester, be very careful and *not copy any code from it*
    - it's best if you just get rid of it
- *Grading guidelines* is the *ONLY* way we will grade and we can only grade on `nunki.usc.edu` in our grading account
- If you make a submission
  - make sure you follow the *"Verify Your Submission"* procedure
- You should start looking for partners for kernel assignments
  - team forming deadline is 2 days after warmup 2 is due
- You need to install Ubuntu 12.04 on your laptop/desktop
  - if there are any problems, I need to know *NOW!*

Housekeeping (Lecture 9 - 2/8,9/2016)

- Recommended timeline for warmup #2
  - get the simulation/emulation to work before this Tuesday
  - add <Cntrl+C> handling code with thread cancellation next week
    - my recommendation is to use `sigwait()` in a signal-catching thread and block SIGINT everywhere else
      - don't use signal handlers!

Housekeeping (Lecture 10 - 2/10,11/2016)

- Warmup #2 due at 11:45pm on Friday, 2/19/2016
  - if you have code from a previous semester, be very careful and *not copy any code from it*
    - it's best if you just get rid of it
- *Grading guidelines* is the *ONLY* way we will grade and we can only grade on `nunki.usc.edu` in our grading account
- If you make a submission
  - make sure you follow the *"Verify Your Submission"* procedure
- New grader to replace Kunul Shah, starting with Warmup #2
  - Hongtai Cao <hongtaic@usc.edu>
- You should start looking for partners for kernel assignments
  - team forming deadline is 2 days after warmup 2 is due
- You need to install Ubuntu 12.04 on your laptop/desktop
  - if there are any problems, I need to know *NOW!*

Housekeeping (Lecture 10 - 2/10,11/2016)

- Recommended timeline for warmup #2
  - get the simulation/emulation to work by now
  - add <Cntrl+C> handling code with thread cancellation next week
    - my recommendation is to use `sigwait()` in a signal-catching thread and block SIGINT everywhere else
      - don't use signal handlers!

Housekeeping (Lecture 11 - 2/16/2016)

- Warmup #2 due at 11:45pm this Friday, 2/19/2016
  - if you have code from a previous semester, be very careful and *not copy any code from it*
    - it's best if you just get rid of it
- *Grading guidelines* is the *ONLY* way we will grade and we can only grade on `nunki.usc.edu` in our grading account
- If you make a submission
  - make sure you follow the *"Verify Your Submission"* procedure
- Kernel team forming deadline is this coming Sunday
  - must follow procedure in the Projects web page
  - I will form random teams starting next Monday
- You need to install Ubuntu 12.04 on your laptop/desktop
  - if there are any problems, I need to know *NOW!*

Housekeeping (Lecture 11 - 2/16/2016)

- You can download the "prestine kernel source" now
  - *save a copy* of the "prestine kernel source" (to be used to Verify Your Kernel Submission later)
  - follow all the instructions
    - make sure everything looks like what the spec says
    - debug the kernel with GDB and make sure it works right
    - if things are not working right, you need to see me (or a TA/CP) as soon as possible
  - read the "weenix documentation"
- You are not expected to be able to do kernel 1 yet
  - *by Wed/Thu next week*, you will know enough
  - the TAs will give an introduction to the kernel assignments this Friday during discussion sections
  - if you are done with warmup #2, feel free to start
    - feel free to ask me questions about kernel 1 (assuming you have read the spec and "weenix documentation")

Housekeeping (Lecture 12 - 2/17,18/2016)

- Warmup #2 due at 11:45pm this Friday, 2/19/2016
  - if you have code from a previous semester, be very careful and *not copy any code from it*
    - it's best if you just get rid of it
- *Grading guidelines* is the *ONLY* way we will grade and we can only grade on `nunki.usc.edu` in our grading account
- If you make a submission
  - make sure you follow the *"Verify Your Submission"* procedure
- Kernel team forming deadline is this coming Sunday
  - must follow procedure in the Projects web page
  - I will form random teams starting next Monday
- You need to install Ubuntu 12.04 on your laptop/desktop
  - if there are any problems, I need to know *NOW!*
- You are not expected to be able to do kernel 1 yet
  - *by Wed/Thu next week*, you will know enough

Housekeeping (Lecture 12 - 2/17,18/2016)

- Things to do this weekend
  - *save a copy* of the "prestine kernel source"
    - you will need it to Verify Your Kernel Submission
  - follow all the instructions
    - make sure everything looks like what the spec says
    - debug the kernel with GDB and make sure it works right
    - if things are not working right, you need to see me (or a TA/CP) as soon as possible
  - *read the kernel assignment web page* and understand all the requirements
    - *especially about grading and testing your kernel (has more requirements than warmups)*
  - read the "weenix documentation"
  - figure out a collaboration strategy with your teammates
    - someone needs to be in charge of documentation and testing
    - someone needs to be in charge of submission and Verifying Your Kernel Submission

Housekeeping (Lecture 13 - 2/22,23/2016)

- Kernel 1 due at 11:45pm on Friday, 3/11/2016
  - if you have code from a previous semester, be very careful and *not copy any code from it*
    - it's best if you just get rid of it
  - read the *kernel FAQ* and starting using `gdb` right away!
  - I'm hoping that by the *end of this week*, I will cover everything you need to complete kernel 1
- *Grading guidelines* is the only way we will grade
  - make sure you have tried everything there
  - remember, we must use the same grading procedure for all
- Please only run `weenix` on Ubuntu 12.04 (14.04 is acceptable)
  - if there are any problems, I need to know now so we can get it resolved *NOW!*
  - don't waste time trying to run it on something else

Housekeeping (Lecture 13 - 2/22,23/2016)

- For kernel 1, you need to write kernel process and thread creation/termination code
  - to see how kernel processes and threads works, read the code in "`proc/faber_test.c`" and "`proc/sunghan_test.c`"
    - you must NOT change a single line in these files
    - you need to write kernel process/thread creation/termination code so that these test code would run perfectly
- Your team need to *meet often*
  - once a day is preferred
    - work at the same place at the same time
    - have lots of discussions (and write a fair amount of code)
  - swallow your pride, be honest with your teammates, don't hide your weakness
    - everyone gets the same grade
    - if no one is really good at this (which is expected), someone (or more) has to step up

Housekeeping (Lecture 13 - 2/22,23/2016)

- You don't have to know what every piece of code is doing
  - learn how to *assume* that other code works (until proven otherwise)
    - other code works kind of like what's covered in lectures
  - use "`grep`" to *get an idea* of how a function is used and how a field in a data structure is used
- It's very important that you *understand every line of code* in `faber_thread_test()`

Housekeeping (Lecture 14 - 2/24,25/2016)

- Kernel 1 due at 11:45pm on Friday, 3/11/2016
  - if you have code from a previous semester, be very careful and *not copy any code from it*
    - it's best if you just get rid of it
- *Grading guidelines* is the only way we will grade
  - make sure you have tried everything there
  - remember, we must use the same grading procedure for all
- When this lecture is finished, you should have everything you need to finish kernel 1
- If you still don't know how to use `gdb`, you have to learn it *NOW*
- Keep `MTP=0` in `Config.mk`
- You should know where every thread is at any time
  - if a thread is *not running*, it must be *sitting in a queue* waiting for something

Housekeeping (Lecture 14 - 2/24,25/2016)

- Kernel 1 implementation timeline
  - till Friday next week: keep `DRIVERS=0` in `Config.mk`
    - get INIT process (with `PID=1`) to start and quit;
    - call `faber_thread_test()` from `initproc_run()`
      - start with `CS402TESTS=1` in `Config.mk`
    - make sure the *kernel halts cleanly*
      - then set `CS402TESTS=2`, `3`, and so on
  - afterwards: set `DRIVERS=1` in `Config.mk`
    - run `kshell` in `initproc_run()`
      - "`help`", "`echo`" and "`exit`" `kshell` commands should work
    - add `kshell` commands to invoke any test function in grading guidelines and your README (see rules about "SELF-checks")
      - for each `kshell` command, you need to create a child process and set the test function as the first procedure of the thread in the child process
  - before you make a submission, make sure there is a way to test/exercise *every code path*

Housekeeping (Lecture 15 - 2/29/2016,3/1/2016)

- Kernel 1 due at 11:45pm on Friday, 3/11/2016
  - if you have code from a previous semester, be very careful and *not copy any code from it*
    - it's best if you just get rid of it
- *Grading guidelines* is the only way we will grade
  - don't change directory structure
  - don't alter or delete first comment block in a `.c` file
  - tests in sections (C), (D), and (E) of the grading guidelines must run in the "foreground"
- If you are confused about something in kernel 1, come to office hours and helpdesk hours *this week*
  - next week may be too late to fix your code!

Housekeeping (Lecture 15 - 2/29/2016,3/1/2016)

- Kernel 1 implementation timeline
  - by this Friday: keep `DRIVERS=0` in `Config.mk`
    - get INIT process (with `PID=1`) to start and quit;
    - call `faber_thread_test()` from `initproc_run()`
      - start with `CS402TESTS=1` (then, 2, 3, ...) in `Config.mk`
    - make sure the *kernel halts cleanly*
  - afterwards: set `DRIVERS=1` in `Config.mk`
    - run `kshell` in `initproc_run()`
      - "`help`", "`echo`" and "`exit`" `kshell` commands should work
    - add `kshell` commands to invoke any test function in grading guidelines and your README (see rules about "SELF-checks")
      - for each `kshell` command, you need to create a child process and set the test function as the first procedure of the thread in the child process
- It's very important that you *understand every line of code* in `faber_thread_test()`

Housekeeping (Lecture 15 - 2/29/2016,3/1/2016)

- Unlike the warmup assignments, if you leave junk in the kernel, you will lose points!
  - the requirement is that there must be a way to test/visit/exercise *every code path you wrote*
  - if a piece of code *you wrote* cannot be visited, just *delete* the code there (thus remove the code path)
  - I would prefer that by running all the tests in (C) and (D) under kshell, every code path you have implemented have been visited
    - section (E) would then be *empty* and you can write, "none needed" in section (E) of the README file

Housekeeping (Lecture 16 - 3/2,3/2016)

- Kernel 1 due at 11:45pm on Friday, 3/11/2016
  - if you have code from a previous semester, be very careful and *not copy any code from it*
    - it's best if you just get rid of it
- *Grading guidelines* is the only way we will grade
- After submission, make sure you *Verify Your Kernel Submission*
  - don't change directory structure
  - don't alter or delete first comment block in a `.c` file
  - tests in sections (C), (D), and (E) of the grading guidelines must run in the "foreground"
- If you are confused about something in kernel 1, come to office hours and helpdesk hours *this week*
  - next week may be too late to fix your code!
- You might want to try GDB assignment 2 as an exercise
  - not graded

Housekeeping (Lecture 16 - 3/2,3/2016)

- Kernel 1 implementation timeline
  - by this Friday: keep `DRIVERS=0` in `Config.mk`
    - get INIT process (with `PID=1`) to start and quit;
    - call `faber_thread_test()` from `initproc_run()`
      - start with `CS402TESTS=1` (then, 2, 3, ...) in `Config.mk`
    - make sure the *kernel halts cleanly*
  - afterwards: set `DRIVERS=1` in `Config.mk`
    - run `kshell` in `initproc_run()`
      - "`help`", "`echo`" and "`exit`" `kshell` commands should work
    - add `kshell` commands to invoke any test function in grading guidelines and your README (see rules about "SELF-checks")
      - for each `kshell` command, you need to create a child process and set the test function as the first procedure of the thread in the child process
- It's very important that you *understand every line of code* in `faber_thread_test()`

Housekeeping (Lecture 16 - 3/2,3/2016)

- I won't be able to respond to every post to the class Google Group
  - if you need an answer from me for a particular question, *forward* a post in a *private e-mail to me*
    - please keep in mind that neither I nor the teaching staff can tell you what code to write
      - but students are *always* welcome to respond to such questions (as long as you don't say it in more than 4 lines of code or pseudo code)
  - to get Google group extra credit, your response needs to be *timely*
    - posted within 48 hours of the original post

Housekeeping (Lecture 17 - 3/7,8/2016)

- Kernel 1 due at 11:45pm this Friday, 3/11/2016
  - if you have code from a previous semester, be very careful and *not copy any code from it*
    - it's best if you just get rid of it
- *Grading guidelines* is the only way we will grade
- After submission, make sure you *Verify Your Kernel Submission*
  - don't change directory structure
  - don't alter or delete first comment block in a `.c` file
  - tests in sections (C), (D), and (E) of the grading guidelines must run in the "foreground"
- If you are confused about "SELF-checks", please come talk to me
- I will go over exam logistics today
  - will post exam coverage on class web site - everything from beginning of semester to first few slides of today's lecture, *minus Ch 5*

Housekeeping (Lecture 18 - 3/9,10/2016)

- Kernel 1 due at 11:45pm this Friday, 3/11/2016
  - if you have code from a previous semester, be very careful and *not copy any code from it*
    - it's best if you just get rid of it
- *Grading guidelines* is the only way we will grade
  - when running `faber_thread_test()`, you need to make sure that all the *exit codes* are correct
    - read the code to figure out what values to expect
  - you should be able to run commands after commands, etc.
  - if you are confused about "SELF-checks", please send me e-mail
- After submission, make sure you *Verify Your Kernel Submission*
  - tests in sections (C), (D), and (E) of the grading guidelines must run in the "foreground"
- This Friday, the TAs will give an introduction to Kernel 2
- By the way, *midterm* exam does cover *kernel 1*