Ecampus Syllabus

Course Name: Operating Systems I
Course Number: CS311
Term Offered: Fall 2013
Credits: 4
Instructor name: R. Jesse Chaney
Instructor email: chaneyr@eecs.orst.edu
Instructor phone: (541) 223-9537 (Google Voice)
Link to instructor bio or website: TBD
Teaching Assistant name and contact info: Bryan Pawlowski (pawlowsb@engr.orst.edu)

Course Description
Introduction to operating systems using POSIX and Unix as the case study. System calls and utilities, fundamentals of processes, threading, and inter-process communication. The course also explores a scripting language, Python.

Prerequisites
CS 261, (CE 271 or CS 271), knowledge of C (you should be very comfortable with programming in C).

Communication
Please post all course-related questions in the Piazza Forum so that the whole class may benefit from your conversation. Please email your instructor for matters of a personal nature. The instructor will reply to course-related questions and email within 24-48 hours.

Any email sent to the instructor about this course **must** originate with an OSU supplied email account and contain the tag [CS311] at the beginning of the subject. Failure to comply with this will result in delayed response to your email.

Technical Assistance
If you experience computer difficulties, need help downloading a browser or plug-in, assistance logging into the course, or if you experience any errors or problems while in your online course, contact the OSU Help Desk for assistance. You can call (541) 737-3474 (USA), email osuhelpdesk@oregonstate.edu or visit the OSU Computer Helpdesk online.

Learning Resources
Required Text: **THE LINUX PROGRAMMING INTERFACE** by Kerrisk ISBN 978-1-59327-220-3 (TLPI)

> **Note to prospective students**: Please check with the OSU Bookstore for up-to-date information for the term you enroll (http://www.osubookstore.com/ or (800) 595-0357 (USA)). If you purchase course materials from other sources, be very careful to obtain the correct ISBN.

Alternative sources for the textbook are: Amazon and the author's web site (note the coupon code for the discount and **ebook,** Mamaku). I can tell you from personal experience that having the ebook is very useful. The search function is very nice.

Additional reading material will be online (Python, PThreads, and additional information on Semaphores). These are all available online without charge.

Supplemental/Recommended reading material is also available online: things such as tutorials for C, make, vi, and emacs.

Blackboard

This course will be delivered via Blackboard where you will interact with your classmates and with your instructor. Within the course Blackboard site you will access the learning materials, such as the syllabus, class discussions, assignments, projects, and quizzes. To preview how an online course works, visit the Ecampus Course Demo. For technical assistance, please visit Ecampus Technical Help.

Measurable Student Learning Outcomes

- Explain why multiprogramming is important for modern operating systems.
- Explain the general structure of a multi-programmed operating system.
- Explain the purpose and operation of system calls.
- Write programs utilizing system calls.
- Write programs using a scripting language (Python).
- Write a program that spawns processes and provides mutual exclusion for variables or other resources shared by the processes.
- Write programs that use inter-process communication: signals, pipes, messages queues, and shared memory.
- Explain how a common file system works, including structure, I/O operations, and security.
- Describe the memory organization of a typical process in a common operating system.

Evaluation of Student Performance

- Quizzes: 10%
- Midterm: 20%
- Final: 30%
- Programming projects: 40%

The grading scale is as follows:

- $90 \leq A \leq 100$
- $80 \leq B < 90$
- $70 \leq C < 80$
- $60 \leq D < 70$
- $00 \leq F < 60$

Other important grading criteria:

- Your exam scores (midterm and final) must have an average ≥ 65 in order to pass the course. If the average of your midterm and final are less than 65, you will receive an F grade.

This course is offered through Oregon State University Extended Campus. For more information, contact:

- You must pass the final exam to pass this class. If you do not receive a passing grade on the final exam, you will receive an F grade.
- All programming projects must be submitted in order to pass the course. Students missing programming projects at the end of the term will automatically receive an F grade.
- Students who do not submit the final exam will automatically receive an F grade.
- Extra credit is available on nearly every assignment in this class (quizzes, exams, and programming assignments). Do not request additional extra credit.
- If you wind up with a grade average or 89.99%, you will get a B in the class. Answering 1 additional question correctly on a quiz would probably have pushed you over to an A. Retaking a quiz may have given you the A.

Quizzes
Quizzes are open book, open note, open Internet, and open lecture. You can use just about anything except your fellow students while taking a quiz. The quizzes will primarily (but not exclusively) cover assigned reading material from the week. Quizzes are not proctored. Quizzes are not timed. You will be able to take each quiz twice. If you are dissatisfied with your result on the initial quiz attempt, you can do some further reading and take the quiz again, the recorded score will be the higher score you receive. Quizzes have a due date. Quizzes cannot be taken after the due date. If you answer a quiz question incorrectly, the section (and or page number) from the book (or online material) will be available to you. You can review that section/page to find the answer and retake the quiz.

Exams
The Midterm and Final are not open book and not open note. You are not allowed to take anything into an exam. Exams are proctored. Exams have time limits. Exams are taken once. Exams have a due date. The Midterm has 60 minutes and the Final has 110 minutes (the maximum allowed). Neither the Midterm nor the Final are intended to be time-pressure tests. A study guide will be made available for each of the exams.

One of the best ways to do well on the quizzes and exams is to take notes while reading the material from the book (or on-line material). For example, create a 1-page summary of each chapter (and don't just copy the summary from the end of the chapter). I know this will slow your reading rate, but you will do better on quizzes, exams, and programming projects. Several students have validated this.

I know taking that many quizzes is not your favorite thing to do. However, I have found that having weekly quizzes helps students keep pace with the work and provides them with quick feedback on how well the material is understood or remembered. Grades from the previous classes validate this. When I do not quiz on a subject, students have done poorly on that subject in the exams.

Programming Projects
The programming projects (homework) are a significant portion of this class. The programming projects are the most common place for students to struggle in this class. Several things about this class' programming projects may be new to you.

- You'll be writing Python programs.
- You will be writing programs in C.
- You will be using a Makefile to compile the program.
- You may be using a debugger that has a basic text based interface.
- You'll be working on Linux.
- You'll be using a lot of system calls.
- You'll be writing complete programs, not just a few functions.
- You'll be parsing the command line.

- You'll be creating new processes.
- You'll be writing multi-threaded applications.
- You'll be using a lot of inter-process communication: pipes, message queues, semaphores, shared memory, and sockets.

All programming projects must be submitted electronically by 11:59pm on the due date at the engineering submission site (TEACH). There is a submission link for each homework assignment.

Late Projects have exactly 7 days (1 week) from the due date, no more, to be submitted. **Late work is penalized 5% per day.** Any programming project submitted more than 7 days after the due date will receive a grade of zero (0).

You have the right to make use of two grace days for submission of programming projects, used in increments of one day. These allow you to have an un-penalized late assignment by up to two days or 2 assignments up to 1 day each. These *must* be invoked prior to the submission deadline, and must be invoked via email to the instructor. Failure to follow this procedure renders the late day invalid, though used. I encourage you to **not** use up your grace days early in the term. Programming assignments get harder as the as the term progresses. You'd hate to waste grace days on early and easy assignments when the assignments get harder. Start your programming assignments as soon as possible. Do not wait until the last weekend to begin them. Watching the clock tick past midnight Monday for an assignment that feels far from working is not enjoyable.

The programming projects are designed to **not** build on each other (with the exception of homework 3/4). This is so that you can limit losses on one assignment and move on to the next one. You can overcome a poor grade on one assignment and still do well in the class. Do not allow struggling on one programming assignment to cause you to be late on all programming assignments. Don't miss submitting a programming assignment. You are much better off to submit a partially functional assignment than to not submit anything for an assignment. As stated above, you must submit **all** programming projects for the class in order to pass the class.

All grading will be done on the Linux system, `eos-class`. You can do your development directly on `eos-class` or you can use your own computer. If you choose to use your own computer for completing the homework assignments. I highly recommend that you do use a Linux operating system or at least test your assignments on `eos-class`. You can log onto `eos-class` via SSH. Standard policies apply: you must have a campus IP to directly log in, or log in via `flip` or `nome`. The `eos-class` server is new this term and was setup specifically for this class (the online class).

I have created a virtual system using VirtualBox that should provide you with an easy way to do homework development on your own computer. The CS311 VM is based on CentOS, a variant of Linux that is very much like the Red Hat running on eos-class. I have also placed the sample source code from TPLI. In addition to the base OS, the CS311 VM includes (as far as I can tell) all packages required to complete the homework assignments for this class. A separate document on the Blackboard site describes the installation and setup for the CS311 VM. There is a version of the CS311 VM both for 64bit systems and for more memory limited 32bit systems. If you try out the CS311 VM and hate it, I'm sorry (a little bit), but still you'll need to find a way to do the homework assignments.

Compilation and running of your projects will be done on `eos-class`. Projects done in C will be compiled with `gcc`. If your project does not compile using `gcc`, on `eos-class`, it does not compile. C programming assignments that do not compile on `eos-class` using `gcc` will receive a grade of 0. Make sure your projects compile, using `gcc`. Even if your C code does nothing functional, make sure it compiles (use a `helloworld.c` file if necessary). Compiling means it produces an executable binary file (object files (`.o` files) are not executable). Compiling with `gcc` does not mean `g++`, `c++`, `C++`, or `C#`. If your program requires I compile it with a `C++` compiler or Visual Studio, I consider that it does not

compile, resulting in a grade of 0. If your assignments compiles fine on your laptop but not on `eos-class`, it is a grade 0.

You must submit your work twice for all homework assignments. Ecampus wants to archive your work through Blackboard and EECS requires you to submit through TEACH to be graded. If you do not submit your assignment through TEACH, it cannot be graded and you will be disappointed with your grade. **Make sure you submit your work through** TEACH. Submit your work for each assignment as a single bzip file through TEACH. It is recommended that you use MS Word/Excel or LibreOffice Writer/Calc for write-ups of the homework (where necessary). Simple README files are best done as plain text files using vi/vim/emacs/nano/eclipse. More detailed instructions on how to combine your files for an assignment can be found on the class Blackboard site. You can submit your assignments more than once through TEACH. Each will be time-stamped. We will grade only the last one submitted.

I make sure you can submit your homework through TEACH, I have created a Homework0 assignment. It does not count any points, but allows you to simply test submitting a file through TEACH. If you cannot submit something through Homework0, you won't be able to submit the other assignments. I recommend you try Homework0 the early the first week.

All homework must be done individually unless specifically allowed to work in groups. All homework must be done individually. The homework programming projects are **not** group projects. You must do your own programming projects. You may share snippets of code but do not share entire source files. You may share pseudo code and ideas about how to solve or approach problems, but write your own code. If you are getting odd compiler messages, you can share the snippet of code that is producing the message; you don't need to share the entire file. All homework must be done individually. There are some very good tools for detecting when students are copying code from each other. I have used those tools. I have found students who copied code. No one enjoyed it. All homework must be done individually.

If you are struggling on a programming assignment, the first thing you should do is make sure you have read the assigned readings for the week and prior weeks and watched the lectures (review the notes you've taken). The book has many excellent descriptions and examples of the topics covered in the programming assignments. The source code from the book is available in expanded form from the author's web site. The lectures are very often additional examples of those same topics. Sending your entire source code to the instructor or TA with a note "Saying something wrong can you fix it?" is unlikely going to get you the response you want. The instructor and TA are not debuggers. Run the program in a debugger yourself (gdb or eclipse). If you identify specific sections in your code (with function names and line numbers), you are much better off, but not guaranteed to get debugging help. The way to get better at debugging code is to use the debuger.

Keys to Success

1) Watch the lectures and take notes (just like you would in an on-campus class)
2) Read the assigned material (and take notes)
3) Take the quizzes (twice if it helps)
4) Start the programming assignments early and complete them on time
5) When struggling with a homework assignment:
    a) Look in the book
    b) Look for man pages
    c) Look on-line
    d) Look for/create a discussion in Piazza
    e) Email the instructor and/or TA
6) The best key to success in this class is: keep up. Don't let yourself fall behind.

One of the skills you'll develop in this class is how to look for things in the resources listed in item 5 (above). You'll spend some quality time with your favorite Internet search engine. You'll learn how to

wade through the chaff of Stack Overflow to find the relevant example from the hundreds of search hits. You'll learn that man pages are a really cool idea. You'll probably be able to remember the page number of certain examples from the textbook. Some of this won't be fun, but you'll learn that you can learn it.

Class Schedule

The planned class schedule is below. While it is possible that we will deviate from the schedule below, I doubt it.

Week	Topics	Reading Assignments	Learning Activities	Due Dates
1	• Overview, setup, and connections. • Installing VirtualBox • Installing CS311 VM • Command line tools • Intro to Python	Python tutorial sections 1-5 Python subprocess module		
2	• Python and shell interaction • Introduction to systems programming concepts • Purpose and operation of system calls • Begin file I/O	Python OS module TLPI 1-4	Quiz #1	Homework #1
3	• File System Interaction • File I/O • File Attributes • POSIX Time	TLPI 5, 15 10	Quiz #2	Homework #2
4	• POSIX Time (continued) • Signals and signal handling • Process creation and control	TLPI 12, 20, 21, 24, 27	Quiz #3	Homework #3
5	• Child process interaction • Pipes and FIFOs • Message Queues	TLPI 25, 26, 44, 46, 52	Quiz #4 **Midterm Exam**	Homework #4
6	• Introduction to threading • Thread synchronization	TLPI 29, 30, 33 (skip 33.5 and 33.6) The PThreads tutorial.	Quiz #5	
7	• Introduction to IPC • POSIX semaphores • POSIX shared memory	TLPI 51, 53, 54	Quiz #6	Homework #5
8	• Sockets • More sockets	TLPI 56, 59 (skip 59.7-59.9)	Quiz #7	
9	• I/O multiplexing • Still more sockets • File monitoring events	TLPI 63.1-2, 19	Quiz #8	Homework #6
10	• Miscellaneous • Review		Quiz #9	Homework #7

	• Dead week			
Finals			**Final Exam**	

Course Policies

Proctored Exams

This course requires that you take exams under the supervision of an approved proctor. Proctoring guidelines and registration for proctored exams are available online through the Ecampus testing and proctoring website. It is important to submit your proctoring request as early as possible to avoid delays.

Makeup Exams

Makeup exams will be given only for missed exams excused in advance by the instructor. Excused absences will not be given for airline reservations, routine illness (colds, flu, stomach aches), or other common ailments. You have 4 days on which to take the Midterm and the Final. Excused absences will generally not be given after the absence has occurred, except under very unusual circumstances.

Exam Time Limits

Exams in this class are timed. If you exceed the time limit on an exam, you will be assessed a penalty of 10% for every fifteen minute interval beyond the time limit. As stated above, it is not the intent of the exams to be time pressured, but pace yourself.

Incompletes

Incomplete (I) grades will be granted only in emergency cases (usually only for a death in the family, major illness or injury, or birth of your child), and if the student has turned in 80% of the points possible (in other words, usually everything but the final exam). If you are having any difficulty that might prevent you completing the coursework, please don't wait until the end of the term; let me know right away. Completion of an incomplete (I) grade will require additional work from you. You won't simply have an opportunity to do the work late; you'll do more.

Statement Regarding Students with Disabilities

Accommodations are collaborative efforts between students, faculty and Disability Access Services (DAS) with accommodations approved through DAS are responsible for contacting the faculty member in charge of the course prior to or during the first week of the term to discuss accommodations. Students who believe they are eligible for accommodations but who have not yet obtained approval through DAS should contact DAS immediately at (541) 737-4098 (USA).

Expectations for Student Conduct

Student conduct is governed by the university's policies, as explained in the Office of Student Conduct: Information and Regulations.

Academic Honesty

Students are expected to do their own work. Individuals are expected to be the sole source of their code. We use software designed to find similarities between programs. Each individual's program is compared to every other individual's program to find similarities. Please do you own work.

Programming assignments present unique challenges for graders. It is often difficult for a grader to distinguish between legitimate help and plagiarism. Therefore, it is sometimes possible to get a good score without really understanding what you have handed in. Understanding is the real point of the class.

Honesty is absolutely essential in order for learning to take place. It will form the foundation of your professional integrity in your career.

If you are having trouble with an assignment, you are encouraged to discuss it with other students, TAs, the instructor, or anyone else who will listen, but don't just have someone else tell you how to solve the problem! If other students ask you for help, don't just let them copy your work! It is possible to discuss problems without plagiarizing. One of the best methods of debugging is to explain your solution to someone else.

If you get help from, give help to, or work together with someone, you must (in the program header block) list that person as a collaborator and describe the help. Programs that are very similar will be subjected to review unless both programs indicate that they were produced collaboratively. We use plagiarism-detection software check your code against the code from other students. It is quite sophisticated and can easily see through variable name changes and formatting differences.

If you get help from printed or online sources, ***you must cite your references***.

If you are found in violation of any of the above policies, whether you are the giver or receiver of help, you will receive a zero on the assignment or fail the course (instructor's discretion). The academic dishonesty charge will be documented and sent to your school's dean and the Office of Student Conduct. The first offense results in a warning; the second offense results in an academic dishonesty charge on your transcript, a disciplinary hearing, and possible expulsion.

The bottom line is: Each student is expected to understand all aspects of the programs s/he submits for credit.

Academic Integrity

Students are expected to comply with all regulations pertaining to academic honesty. For further information, visit Avoiding Academic Dishonesty, or contact the office of Student Conduct and Mediation at 541-737-3656.

OAR 576-015-0020 (2) Academic or Scholarly Dishonesty:

a) Academic or Scholarly Dishonesty is defined as an act of deception in which a Student seeks to claim credit for the work or effort of another person, or uses unauthorized materials or fabricated information in any academic work or research, either through the Student's own efforts or the efforts of another.

b) It includes:

(i) CHEATING - use or attempted use of unauthorized materials, information or study aids, or an act of deceit by which a Student attempts to misrepresent mastery of academic effort or information. This includes but is not limited to unauthorized copying or collaboration on a test or assignment, using prohibited materials and texts, any misuse of an electronic device, or using any deceptive means to gain academic credit.

(ii) FABRICATION - falsification or invention of any information including but not limited to falsifying research, inventing or exaggerating data, or listing incorrect or fictitious references.

(iii) ASSISTING - helping another commit an act of academic dishonesty. This includes but is not limited to paying or bribing someone to acquire a test or assignment, changing someone's grades or academic records, taking a test/doing an assignment for someone else by any means, including misuse of an electronic device. It is a violation of Oregon state law to create and offer to sell part or all of an educational assignment to another person (ORS 165.114).

(iv) TAMPERING - altering or interfering with evaluation instruments or documents.

This course is offered through Oregon State University Extended Campus. For more information, contact:
Web: ecampus.oregonstate.edu Email: ecampus@oregonstate.edu Tel: 800-667-1465

(v) PLAGIARISM - representing the words or ideas of another person or presenting someone else's words, ideas, artistry or data as one's own, or using one's own previously submitted work. Plagiarism includes but is not limited to copying another person's work (including unpublished material) without appropriate referencing, presenting someone else's opinions and theories as one's own, or working jointly on a project and then submitting it as one's own.

c) Academic Dishonesty cases are handled initially by the academic units, following the process outlined in the University's Academic Dishonesty Report Form, and will also be referred to SCCS for action under these rules.

Conduct in this Online Classroom
Students are expected to conduct themselves in the course (e.g., on discussion boards, email postings) in compliance with the university's regulations regarding civility.

Tutoring
NetTutor is a leading provider of online tutoring and learner support services fully staffed by experienced, trained and monitored tutors. Students connect to live tutors from any computer that has Internet access. NetTutor provides a virtual whiteboard that allows tutors and students to work on problems in a real time environment. They also have an online writing lab where tutors critique and return essays within 24 to 48 hours. Access NetTutor from within your Blackboard class by clicking on the Tools button in your course menu.

OSU Student Evaluation of Teaching
Course evaluation results are extremely important and are used to help me improve this course and the learning experience of future students. Results from the multiple choice questions are tabulated anonymously and go directly to instructors and department heads. Student comments on the open-ended questions are compiled and confidentially forwarded to each instructor, per OSU procedures. The online Student Evaluation of Teaching form will be available toward the end of each term, and the Office of Academic Programs, Assessment, and Accreditation will send you instructions via your ONID email address. You will log in to “Student Online Services” to respond to the online questionnaire. The results on the form are anonymous and are not tabulated until after grades are posted.


deadlock
GNU
stack
listen
jiffy
join
uname
send
fstat
queue
perror
socket
debug
access
string
select
malloc
parent
receive
chmod
epoll
memory
realtime
exception
orphan
Makefile
semaphore
SIGCHLD
network
Linux
zombie
POSIX
ctime
root
Unix
fork
synchronization
argc
dup2
message
gid
pid
reentrant
close
getopt
heap
thread
argv
reap
bytes
wait
pause
kill
atomicity
write
script
read
signal
kernel
process
errno
mutex
text
block
chown
uid