The Unix Workbench

Johns Hopkins University

About this Course
Unix forms a foundation that is often very helpful for accomplishing other goals you might have for you and your computer, whether that goal is running a business, writing a book, curing disease, or creating the next great app. The means to these goals are sometimes carried out by writing software. Software can't be mined out of the ground, nor can software seeds be planted in spring to harvest by autumn. Software isn't produced in factories on an assembly line. Software is a hand-made, often bespoke good. If a software developer is an artisan, then Unix is their workbench. Unix provides an essential and simple set of tools in a distraction-free environment. Even if you're not a software developer learning Unix can open you up to new methods of thinking and novel ways to scale your ideas.

This course is intended for folks who are new to programming and new to Unix-like operating systems like macOS and Linux distributions like Ubuntu. Most of the technologies discussed in this course will be accessed via a command line interface. Command line interfaces can seem alien at first, so this course attempts to draw parallels between using the command line and actions that you would normally take while using your mouse and keyboard. You'll also learn how to write little pieces of software in a programming language called Bash, which allows you to connect together the tools we'll discuss. My hope is that by the end of this course you be able to use different Unix tools as if they're interconnecting Lego bricks.

Show less

Language	Volunteer to translate subtitles for this course
How To Pass	Pass all graded assignments to complete the course.
User Ratings	4.6 stars

Syllabus

WEEK 1

Unix and Command Line Basics

This week we'll help you get access to Unix (you may already be using it), and you'll start using the command line. We'll draw parallels between using your mouse and keyboard with your computer's graphics versus only using the command line.

1 video, 13 readings

1. **Video:** Welcome to Week 1
2. **Reading:** Introduction
3. **Reading:** The Unix Workbench Book
4. **Reading:** What is Unix?
5. **Reading:** Mac & Ubuntu Users
6. **Reading:** Windows
7. **Reading:** Hello Terminal!
8. **Reading:** Hello Terminal! Exercises
9. **Reading:** Navigating the Command Line
10. **Reading:** Navigating the Command Line Exercises
11. **Reading:** Creation and Inspection
12. **Reading:** Creation and Inspection Exercises
13. **Reading:** Migration and Destruction
14. **Reading:** Migration and Destruction Exercises

Show less

Graded: Command Line Basics

WEEK 2

Working with Unix

Now we'll get into the power of different Unix tools. We'll walk through several scenarios where you could use Unix to perform tasks at a much faster speed than you would be able to normally.

1 video, 16 readings

1. **Video:** Welcome to Week 2

2. **Reading:** Self-Help
3. **Reading:** Self-Help Exercises
4. **Reading:** Get Wild
5. **Reading:** Get Wild Exercises
6. **Reading:** Regular Expressions
7. **Reading:** Metacharacters
8. **Reading:** Character Sets
9. **Reading:** Escaping, Anchors, Odds, and Ends
10. **Reading:** Find
11. **Reading:** Search Exercises
12. **Reading:** History
13. **Reading:** Customizing Bash
14. **Reading:** Differentiate
15. **Reading:** Pipes
16. **Reading:** Pipes Exercises
17. **Reading:** Make

Show less

Graded: Working with Unix

WEEK 3

Bash Programming

During this week we'll unleash the command line's usefulness as a programming language. By the end of this week you'll be writing your own little computer programs that you can use on the command line.

1 video, 25 readings

1. **Video:** Welcome to Week 3
2. **Reading:** Math
3. **Reading:** Math Exercises
4. **Reading:** Variables

5. **Reading:** Variables Exercises
6. **Reading:** User Input
7. **Reading:** User Input Exercise
8. **Reading:** Conditional Execution
9. **Reading:** Conditional Expressions
10. **Reading:** If and Else
11. **Reading:** Logic and If/Else Exercises
12. **Reading:** Arrays
13. **Reading:** Arrays Exercises
14. **Reading:** Braces
15. **Reading:** Braces Exercise
16. **Reading:** for
17. **Reading:** while
18. **Reading:** Nesting
19. **Reading:** Loops Exercises
20. **Reading:** Writing Functions
21. **Reading:** Getting Values from Functions
22. **Reading:** Functions Exercises
23. **Reading:** The Unix Philosophy
24. **Reading:** Making Programs Executable
25. **Reading:** Environmental Variables
26. **Reading:** Writing Programs Exercises

Show less

Graded: Bash Programming

WEEK 4

Git and GitHub

First you'll learn how to use Git, which is like "track changes" for your code and plain text files, but much more powerful. We'll then explore how to use Git with GitHub, a social coding network

where you can publish you projects and explore other's code.

1 video, 16 readings

1. **Video:** Welcome to Week 4
2. **Reading:** What are Git and GitHub?
3. **Reading:** Setting Up Git and GitHub
4. **Reading:** Getting Started with Git
5. **Reading:** Git Exercises
6. **Reading:** Gitting Help, Logs, and Diffs
7. **Reading:** Ignoring Files
8. **Reading:** Important Git Features Exercises
9. **Reading:** Branching, Part 1
10. **Reading:** Branching, Part 2
11. **Reading:** Branching Exercises
12. **Reading:** GitHub
13. **Reading:** Markdown
14. **Reading:** Pull Requests
15. **Reading:** Pages
16. **Reading:** Forking
17. **Reading:** GitHub Exercises

Show less

Graded: Git & GitHub

Graded: Bash, Make, Git, and GitHub

Nephology

Finally we'll set up a cloud computing environment so we can explore how computers communicate with each other using the internet.

11 readings expand

1. **Reading:** Introduction to Cloud Computing

Graded: Nephology

2. **Reading:** Setting Up DigitalOcean
3. **Reading:** Connecting to the Cloud

View Less

4. **Reading:** Moving Files In and Out of the Cloud
5. **Reading:** Talking to Other Servers
6. **Reading:** Automating Tasks
7. **Reading:** Cloud Computing Exercises
8. **Reading:** Shutting Down a Server
9. **Reading:** Next Steps
10. **Reading:** Giving Feedback
11. **Reading:** Using This Book

How It Works

GENERAL

How do I pass the course?

To earn your course certificate, you'll need to earn a passing grade on each of the required assignments—these can be quizzes, peer-graded assignments, or programming assignments. Videos, readings, and practice exercises are there to help you prepare for the graded assignments.

More

What do start dates and end dates mean?

Courses on Coursera are offered that run multiple times a year — each with a specific start and end date. Once you enroll, you'll have access to all videos, readings, quizzes, and programming assignments (if applicable). Peer-graded assignments can only be submitted and reviewed once your session has begun. If you choose to explore the course without purchasing, you may not be able to access certain assignments. If you don't finish all graded assignments before the end of the session, you can enroll in the next session. Your progress will be saved and you'll be able to pick up where you left off when the next session begins.

More

What are due dates? Is there a penalty for submitting my work after a due date?

Within each session there are suggested due dates to help you manage your schedule and keep coursework from piling up. Quizzes and programming assignments can be submitted late without consequence. However, it is possible that you won't receive a grade if you submit your peer-graded assignment too late because classmates usually review assignment within three days of the assigned deadline.

View the course in catalog

PEER-GRADED ASSIGNMENTS

Peer-graded assignments require you and your classmates to grade each other's work.

How do peer graded assignments work?

After you submit your work, you'll be asked to review your classmates' assignments. To pass, you'll need to earn a passing grade on your submission and complete the required number of reviews.

How are grades calculated?

You and your classmates will be asked to provide a score for each part of the assignment. Final grades are calculated by combining the median scores you received for each section.

What kind of feedback should I give?

Be [illegible] encouraging and honest. Acknowledge what your classmate did well and offer specific sug[illegible]ow they can improve. Scores should reflect the learner's understanding of the assi[illegible]pt and points should not be deducted for difficulties with language or differences in opi[illegible]

Can I re-attempt an assignment?

Yes, [illegible] improve your grade, you can always try again. If you're re-attempting a peer-graded assi[illegible]ubmit your work as soon as you can to make sure there's enough time for your class[illegible]view your work. In some cases you may need to wait before re-submitting a programming assignment or quiz. We encourage you to review course material during this delay.

Is there a penalty for submitting my work late?

No, [illegible]tant to submit your work as close to the due date as you can. Classmates grade most of th[illegible]ts within three days of the due date. If you submit yours too late, there may not be any [illegible] your work.

If I fail an assignment, can I try again?

Related Courses

Concurrent Programming in Java

Distributed Programming in Java

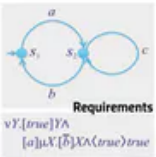
Rice University

Parallel Programming in Java

Yes,  ways try again, but you'll need to resubmit your work as soon as possible to make sure ates have enough time to grade your work.

Can signment?

Yes, but you'll need to re-submit your work and any grade you've already received will be deleted.



System Validation (3): Requirements by modal formulas

EIT Digital

System Validation (2): Model process behaviour

EIT Digital