Cal State Fullerton

# CPSC 254

## Software Development With Open Source Systems - Version Control

Instructor: Tejaas Mukunda Reddy

# Version Control Best Practices

### Commit Related Changes

- A commit should be a wrapper for related changes. For example, fixing two different bugs should produce two separate commits. Small commits make it easier for other team members to understand the changes and roll them back if something went wrong. With tools like the staging area and the ability to stage only parts of a file, Git makes it easy to create very granular commits.

### Commit Often

- Committing often keeps your commits small and, again, helps you commit only related changes. Moreover, it allows you to share your code more frequently with others. That way it's easier for everyone to integrate changes regularly and avoid having merge conflicts. Having few large commits and sharing them rarely, in contrast, makes it hard both to solve conflicts and to comprehend what happened.

### Don't Commit Half-Done Work

- You should only commit code when it's completed. This doesn't mean you have to complete a whole, large feature before committing. Quite the contrary: split the feature's implementation into logical chunks and remember to commit early and often. But don't commit just to have something in the repository before leaving the office at the end of the day. If you're tempted to commit just because you need a clean working copy (to check out a branch, pull in changes, etc.) consider using Git's "Stash" feature instead.

### Test Before You Commit

- Resist the temptation to commit something that you "think" is completed. Test it thoroughly to make sure it really is completed and has no side effects (as far as one can tell). While committing half-baked things in your local repository only requires you to forgive yourself, having your code tested is even more important when it comes to pushing / sharing your code with others.

Prepared By: Tejaas Mukunda Reddy, Shivansh Vijay Nathan

Reference Prof. David Heckathorn

# Version Control Best Practices

### Write Good Commit Messages

- Begin your message with a short summary of your changes (up to 50 characters as a guideline). Separate it from the following body by including a blank line. The body of your message should provide detailed answers to the following questions: What was the motivation for the change? How does it differ from the previous implementation? Use the imperative, present tense („change“, not „changed“ or „changes“) to be consistent with generated messages from commands like git merge.

### Version Control is not a Backup System

- Having your files backed up on a remote server is a nice side effect of having a version control system. But you should not use your VCS like it was a backup system. When doing version control, you should pay attention to committing semantically (see “related changes”) – you shouldn’t just cram in files.

### Use Branches

- Branching is one of Git’s most powerful features – and this is not by accident: quick and easy branching was a central requirement from day one. Branches are the perfect tool to help you avoid mixing up different lines of development. You should use branches extensively in your development workflows: for new features, bug fixes, experiments, ideas…

### Agree on a Workflow

- Git lets you pick from a lot of different workflows: long-running branches, topic branches, merge or rebase, git-flow… Which one you choose depends on a couple of factors: your project, your overall development and deployment workflows and (maybe most importantly) on your and your teammates’ personal preferences. However you choose to work, just make sure to agree on a common workflow that everyone follows.

Prepared By: Tejaas Mukunda Reddy, Shivansh Vijay Nathan

Reference Prof. David Heckathorn

Cal State Fullerton

# CPSC 254

## Software Development With Open Source Systems - Open Source Libraries

Instructor: Tejaas Mukunda Reddy

# Basis of Open Source

What is "Open Source" software?

- Generally, Open Source software is software that can be freely accessed, used, changed, and shared (in modified or unmodified form) by anyone. Open source software is made by many people, and distributed under licenses that comply with the Open Source Definition.
- The internationally recognized Open Source Definition provides ten criteria that must be met for any software license, and the software distributed under that license, to be labeled "Open Source software." Only software licensed under an OSI-approved Open Source license should be labeled "Open Source" software.

Can Open Source software be used for commercial purposes?

- Absolutely. All Open Source software can be used for commercial purpose; the Open Source Definition guarantees this. You can even sell Open Source software.
- However, note that commercial is not the same as proprietary. If you receive software under an Open Source license, you can always use that software for commercial purposes, but that doesn't always mean you can place further restrictions on people who receive the software from you. In particular, copyleft-style Open Source licenses require that, in at least some cases, when you distribute the software, you must do so under the same license you received it under.

Can I restrict how people use an Open Source licensed program?

- No. The freedom to use the program for any purpose is part of the Open Source Definition. Open source licenses do not discriminate against fields of endeavor.

Can I stop "evil people" from using my program?

- No. The Open Source Definition specifies that Open Source licenses may not discriminate against persons or groups. Giving everyone freedom means giving evil people freedom, too.

Prepared By: Tejaas Mukunda Reddy, Shivansh Vijay Nathan

Reference Prof. David Heckathorn

# What is free software is it same as open source

- "Free software" and "open source software" are two terms for the same thing: software released under licenses that guarantee a certain specific set of freedoms.
- The term "free software" is older, and is reflected in the name of the Free Software Foundation (FSF), an organization founded in 1985 to protect and promote free software. The term "open source" was coined by Christine Peterson and adopted in 1998 by the founders of the Open Source Initiative. Like the FSF, the OSI's founders supported the development and distribution of free software, but they disagreed with the FSF about how to promote it, believing that software freedom was primarily a practical issue rather than an ideological one (see for example the entry "How is `open source' related to `free software'?" from the OSI's original 1998 FAQ page).
- Many who later adopted the term "open source" broadly shared the ideological perspective of the FSF but had some disagreements over strategy and rhetoric. Today some people use both terms, choosing according to context and audience.
- One of the tactical concerns often cited by adopters of the term "open source" was the ambiguity of the English word "free", which can refer either to freedom or to mere monetary price; this ambiguity was also given by the OSI founders as a reason to prefer the new term (see "What Does `free' Mean, Anyway?", and similar language on the marketing for hackers page, both from the original 1998 web site).
- In the 1990s, the term "open" applied to software source code was sometimes used to imply source code being merely inspectable or visible or available. Going back further, in the 1980s there were uses of "open" in the computing industry that primarily connoted something like "absence of hardware vendor lockin". OSI's term "open source", as defined in the Open Source Definition, makes clear that open source specifically entails not mere inspection access but also conveying to recipients the perpetual right to fork covered code and use it without additional fees.

Prepared By: Tejaas Mukunda Reddy, Shivansh Vijay Nathan

Reference Prof. David Heckathorn

Cal State Fullerton

# CPSC 254

## Software Development With Open Source Systems - Project Development

Instructor: Tejaas Mukunda Reddy

# Agile

- Executive Brief, the technology management resource for business leaders, offers proven tips, techniques, and action plans that companies can use to better manage people, processes and tools - the keys to improving their business performance. To learn more, please visit: SoftServe United Blog
- © Executive Brief 2008 https://www.projectsmart.co.uk/9-steps-to-a-hassle-free-and-effective-software-development-project.php

**Agile software development**

- Agile software development describes an approach to software development under which requirements and solutions evolve through the collaborative effort of self-organizing and cross-functional teams and their customer(s)/end users(s).[1] It advocates adaptive planning, evolutionary development, early delivery, and continual improvement, and it encourages rapid and flexible response to change.[2]
- The term agile (sometimes written Agile)[3] was popularized, in this context, by the Manifesto for Agile Software Development.[4] The values and principles espoused in this manifesto were derived from and underpin a broad range of software development frameworks, including Scrum and Kanban.[5][6]
- While there is significant anecdotal evidence that adopting agile practices and values improves the agility of software professionals, teams and organizations; some empirical studies have found no scientific evidence.

- **Pair programming**, an agile development technique used by XP. Note information radiators in the background.
- Iterative, incremental and evolutionary: Most agile development methods break product development work into small increments that minimize the amount of up-front planning and design. Iterations, or sprints, are short time frames (timeboxes) that typically last from one to four weeks. Each iteration involves a cross-functional team working in all functions: planning, analysis, design, coding, unit testing, and acceptance testing. At the end of the iteration a working product is demonstrated to stakeholders. This minimizes overall risk and allows the product to adapt to changes quickly.[22] An iteration might not add enough functionality to warrant a market release, but the goal is to have an available release (with minimal bugs) at the end of each iteration.[23] Multiple iterations might be required to release a product or new features.
- Working software is the primary measure of progress.
- **Efficient and face-to-face communication:** No matter which development method is followed, every team should include a customer representative (Product Owner in Scrum). This person is agreed by stakeholders to act on their behalf and makes a personal commitment to being available for developers to answer questions throughout the iteration. At the end of each iteration, stakeholders and the customer representative review progress and re-evaluate priorities with a view to optimizing the return on investment(ROI) and ensuring alignment with customer needs and company goals.
- In agile software development, an information radiator is a (normally large) physical display located prominently near the development team, where passers-by can see it. It presents an up-to-date summary of the product development status.[24][25] A build light indicator may also be used to inform a team about the current status of their product development.
- Very short feedback loop and adaptation cycle
- A common characteristic in agile software development is the daily stand-up (also known as the daily scrum). In a brief session, team members report to each other what they did the previous day toward their team's iteration goal, what they intend to do today toward the goal, and any roadblocks or impediments they can see to the goal.
- **Quality focus:** Specific tools and techniques, such as continuous integration, automated unit testing, pair programming, test-driven development, design patterns, behavior-driven development, domain-driven design, code refactoring and other techniques are often used to improve quality and enhance product development agility. The idea is that the quality is built into the software.
- **Philosophy:** Compared to traditional software engineering, agile software development mainly targets complex systems and product development with dynamic, non-deterministic and non-linear characteristics. Accurate estimates, stable plans, and predictions are often hard to get in early stages, and confidence in them is likely to be low. Agile practitioners will seek to reduce the "leap-of-faith" that is needed before any evidence of value can be obtained.[29] Requirements and design are held to be emergent. Big up-front specifications would probably cause a lot of waste in such cases, i.e., are not economically sound. These basic arguments and previous industry experiences, learned from years of successes and failures, have helped shape agile development's favor of adaptive, iterative and evolutionary development.[30]

Prepared By: Tejaas Mukunda Reddy, Shivansh Vijay Nathan

Reference Prof. David Heckathorn

- **Adaptive vs. predictive**
- Development methods exist on a continuum from adaptive to predictive.[31] Agile software development methods lie on the adaptive side of this continuum. One key of adaptive development methods is a "Rolling Wave" approach to schedule planning, which identifies milestones but leaves flexibility in the path to reach them, and also allows for the milestones themselves to change.[32]
- Adaptive methods focus on adapting quickly to changing realities. When the needs of a project change, an adaptive team changes as well. An adaptive team has difficulty describing exactly what will happen in the future. The further away a date is, the more vague an adaptive method is about what will happen on that date. An adaptive team cannot report exactly what tasks they will do next week, but only which features they plan for next month. When asked about a release six months from now, an adaptive team might be able to report only the mission statement for the release, or a statement of expected value vs. cost.
- Predictive methods, in contrast, focus on analyzing and planning the future in detail and cater for known risks. In the extremes, a predictive team can report exactly what features and tasks are planned for the entire length of the development process. Predictive methods rely on effective early phase analysis and if this goes very wrong, the project may have difficulty changing direction. Predictive teams often institute a change control board to ensure they consider only the most valuable changes.
- Risk analysis can be used to choose between adaptive (agile or value-driven) and predictive (plan-driven) methods.[33] Barry Boehm and Richard Turner suggest that each side of the continuum has its own home ground, as follows:
- **Iterative vs. waterfall**
- One of the differences between agile software development methods and waterfall is the approach to quality and testing. In the waterfall model, there is always a separate testing phase after a build phase; however, in agile software development testing is completed in the same iteration as programming.
- Because testing is done in every iteration—which develops a small piece of the software—users can frequently use those new pieces of software and validate the value. After the users know the real value of the updated piece of software, they can make better decisions about the software's future. Having a value retrospective and software re-planning session in each iteration—Scrum typically has iterations of just two weeks—helps the team continuously adapt its plans so as to maximize the value it delivers. This follows a pattern similar to the PDCA cycle, as the work is planned, done, checked (in the review and retrospective), and any changes agreed are acted upon.
- This iterative approach supports a product rather than a project mindset. This provides greater flexibility throughout the development process; whereas on projects the requirements are defined and locked down from the very beginning, making it difficult to change them later. Iterative product development allows the software to evolve in response to changes in business environment or market requirements.[35]
- Because of the short iteration style of agile software development, it also has strong connections with the lean startup concept.

- **Code vs. Documentation**
- In a letter to IEEE Computer, Steven Rakitin expressed cynicism about agile software development, calling it "yet another attempt to undermine the discipline of software engineering" and translating "Working software over comprehensive documentation" as "We want to spend all our time coding. Remember, real programmers don't write documentation."[36]
- This is disputed by proponents of agile software development, who state that developers should write documentation if that is the best way to achieve the relevant goals, but that there are often better ways to achieve those goals than writing static documentation.[37] Scott Ambler states that documentation should be "Just Barely Good Enough" (JBGE),[38] that too much or comprehensive documentation would usually cause waste, and developers rarely trust detailed documentation because it's usually out of sync with code,[37] while too little documentation may also cause problems for maintenance, communication, learning and knowledge sharing. Alistair Cockburn wrote of the Crystal Clear method:
- Crystal considers development a series of co-operative games, and intends that the documentation is enough to help the next win at the next game. The work products for Crystal include use cases, risk list, iteration plan, core domain models, and design notes to inform on choices...however there are no templates for these documents and descriptions are necessarily vague, but the objective is clear, just enough documentation for the next game. I always tend to characterize this to my team as: what would you want to know if you joined the team tomorrow.
- Alistair Cockburn.
- Agile software development methods
- Agile software development methods support a broad range of the software development life cycle.[40] Some focus on the practices (e.g., XP, pragmatic programming, agile modeling), while some focus on managing the flow of work (e.g., Scrum, Kanban). Some support activities for requirements specification and development (e.g., FDD), while some seek to cover the full development life cycle (e.g., DSDM, RUP).

**Kanban (development)**

- From Wikipedia, the free encyclopedia
- This article is about the process-management and improvement method. For the lean-manufacturing process, see Kanban. Kanban is a lean method to manage and improve work across human systems. This approach aims to manage work by balancing the demands with available capacity, and improving the handling of system level bottlenecks.
- Work items are visualized to give participants a view of progress and process, from start to finish usually via a Kanban board. Work is pulled as capacity permits, rather than work being pushed into the process when requested.
- In knowledge work and software development, this provides a visual process management system which aids decision-making about what, when and how much to produce. Although the underlying Kanban method originated in lean manufacturing[1] (inspired by the Toyota Production System[2]) it is used mainly for software development and technology related work. However Kanban can be applied to any area of work, and it can even be combined with other methods or frameworks such as Scrum.[3]
- https://en.wikipedia.org/wiki/Kanban_(development)

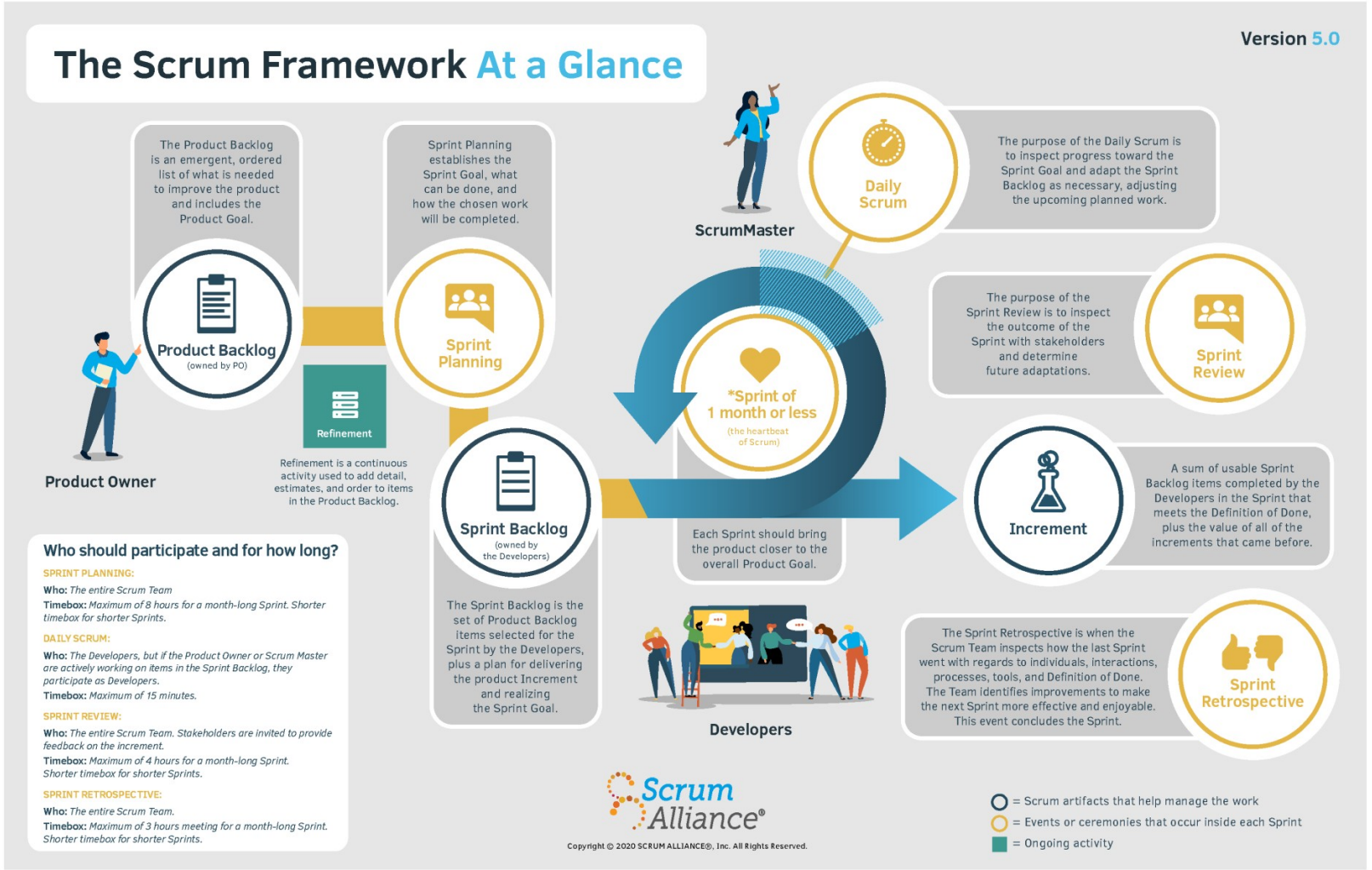
- **"RAD tool"** and **"Rapid Application Development Tool"** redirect here. For development tools focused on making graphical user interfaces, see graphical user interface builder. Rapid-application development (RAD) is both a general term, used to refer to adaptive software development approaches, as well as the name for James Martin's approach to rapid development. In general, RAD approaches to software development put less emphasis on planning and more emphasis on an adaptive process. Prototypes are often used in addition to or sometimes even in place of design specifications.
- RAD is especially well suited for (although not limited to) developing software that is driven by user interface requirements. Graphical user interface builders are often called rapid application development tools. Other approaches to rapid development include the adaptive, agile, spiral, and unified models.
- https://en.wikipedia.org/wiki/Rapid_application_development

**Scrum (software development)**

- Scrum is an agile framework for managing work with an emphasis on software development. It is designed for teams of three to nine developers who break their work into actions that can be completed within timeboxed iterations, called sprints (typically two-weeks) and track progress and re-plan in 15-minute stand-up meetings, called daily scrums.[1][2] Approaches to coordinating the work of multiple scrum teams in larger organizations include Large-Scale Scrum, Scaled Agile Framework (SAFe) and Scrum of Scrums, among others.

**Key ideas:**

- Scrum is an iterative and incremental framework for managing product development.[3][4] It defines "a flexible, holistic product development strategy where a development team works as a unit to reach a common goal",[5] challenges assumptions of the "traditional, sequential approach"[5] to product development, and enables teams to self-organize by encouraging physical co-location or close online collaboration of all team members, as well as daily face-to-face communication among all team members and disciplines involved.
- A key principle of Scrum is the dual recognition that customers will change their minds about what they want or need (often called requirements volatility[6]) and that there will be unpredictable challenges—for which a predictive or planned approach is not suited. As such, Scrum adopts an evidence-based empirical approach—accepting that the problem cannot be fully understood or defined up front, and instead focusing on how to maximize the team's ability to deliver quickly, to respond to emerging requirements, and to adapt to evolving technologies and changes in market conditions.
- Many of the terms used in Scrum (e.g., scrum master) are typically written with leading capitals (i.e., Scrum Master) or as conjoint words written in camel case(i.e., ScrumMaster). To maintain an encyclopedic tone, however, this article uses normal sentence case for these terms—unless they are recognized marks (such as Certified Scrum Master). This is occasionally seen written in all-capitals, as SCRUM.[7] The word is not an acronym, so this is not correct; however, it likely arose due to an early paper by Ken Schwaber which capitalized SCRUM in its title.
- While the trademark on the term Scrum itself has been allowed to lapse, so that it is deemed as owned by the wider community rather than an individual,[9] the leading capital is retained—except when used with other words (as in daily scrum or scrum team). - https://en.wikipedia.org/wiki/Scrum_(software_development)




Prepared By: Tejaas Mukunda Reddy, Shivansh Vijay Nathan

Reference Prof. David Heckathorn

# Pair programming

- Two co-workers pair programming
- Pair programming is an agile software development technique in which two programmers work together at one workstation. One, the driver, writes code while the other, the observer or navigator,[1] reviews each line of code as it is typed in. The two programmers switch roles frequently.
- While reviewing, the observer also considers the "strategic" direction of the work, coming up with ideas for improvements and likely future problems to address. This is intended to free the driver to focus all of their attention on the "tactical" aspects of completing the current task, using the observer as a safety net and guide.

**Economics**

- Pair programming increases the man-hours required to deliver code compared to programmers working individually. Experiments yielded diverse results, suggesting increases of between 15% and 100%.[2] However, the resulting code has about 15% fewer defects.[3] Along with code development time, other factors like field support costs and quality assurance also figure in to the return on investment. Pair programming might theoretically offset these expenses by reducing defects in the programs.
- Design quality: A system with two programmers possesses greater potential for the generation of more diverse solutions to problems for three reasons:
    - the programmers bring different prior experiences to the task;
    - they may access information relevant to the task in different ways;
    - they stand in different relationships to the problem by virtue of their functional roles.
- In an attempt to share goals and plans, the programmers must overtly negotiate a shared course of action when a conflict arises between them. In doing so, they consider a larger number of ways of solving the problem than a single programmer alone might do. This significantly improves the design quality of the program as it reduces the chances of selecting a poor method.[4]
- Satisfaction: In an online survey of pair programmers, 96% of them stated that they enjoyed their work more than when they programmed alone and 95% said that they were more confident in their solutions when they pair programmed.[5]
- Learning: Knowledge is constantly shared between pair programmers, whether in the industry or in a classroom, many sources suggest that students show higher confidence when programming in pairs,[5] and many learn whether it be from tips on programming language rules to overall design skill.[6] In "promiscuous pairing", each programmer communicates and works with all the other programmers on the team rather than pairing only with one partner, which causes knowledge of the system to spread throughout the whole team.[3] Pair programming allows programmers to examine their partner's code and provide feedback which is necessary to increase their own ability to develop monitoring mechanisms for their own learning activities.[6]

Prepared By: Tejaas Mukunda Reddy, Shivansh Vijay Nathan

Reference Prof. David Heckathorn

## Pair Programming

**Team-building and communication**

- Pair programming allows team members to share problems and solutions quickly making them less likely to have hidden agendas from each other. This helps pair programmers to learn to communicate more easily. “This raises the communication bandwidth and frequency within the project, increasing overall information flow within the team.”

**Studies:** There are both empirical studies and meta-analyses of pair programming. The empirical studies tend to examine the level of productivity and the quality of the code, while meta-analyses may focus on biases introduced by the process of testing and publishing.

- A meta-analysis found pairs typically consider more design alternatives than programmers working alone, arrive at simpler more maintainable designs, and catch design defects earlier. However, it raised concerns its findings may have been influenced by "signs of publication bias among published studies on pair programming". It concluded that "pair programming is not uniformly beneficial or effective".[7]
- Although pair programmers may complete a task faster than a solo programmer, the total number of man-hours increases.[2] A manager would have to balance faster completion of the work and reduced testing and debugging time against the higher cost of coding. The relative weight of these factors can vary by project and task.
- The benefit of pairing is greatest on tasks that the programmers do not fully understand before they begin: that is, challenging tasks that call for creativity and sophistication, and for novices as compared to experts.[2] Pair programming could be helpful for attaining high quality and correctness on complex programming tasks, but it would also increase the development effort (cost) significantly.[7]
- On simple tasks, which the pair already fully understands, pairing results in a net drop in productivity.[2][8] It may reduce the code development time but also risks reducing the quality of the program.[7] Productivity can also drop when novice–novice pairing is used without sufficient availability of a mentor to coach them.[9]

Indicators of non-performance: There are indicators that a pair is not performing well:

- Disengagement may present as one of the members physically withdraws away from the keyboard, accesses email, or even falls asleep.
- The "Watch the Master" phenomenon can arise if one member is more experienced than the other. In this situation, the junior member may take the observer role, deferring to the senior member of the pair for the majority of coding activity. This can easily lead to disengagement.

Prepared By: Tejaas Mukunda Reddy, Shivansh Vijay Nathan

Reference Prof. David Heckathorn

Cal State Fullerton

# CPSC 254

## Software Development With Open Source Systems - Linux Utilities

Instructor: Tejaas Mukunda Reddy

# Linux like Unix comes equipped with a number of useful utilities

**grep**

- https://www.gnu.org/software/grep/manual/grep.html
- grep searches input files for lines containing a match to a given pattern list. When it finds a match in a line, it copies the line to standard output (by default) or produces whatever other sort of output you have requested with options.
- Though grep expects to do the matching on text, it has no limits on input line length other than available memory, and it can match arbitrary characters within a line. If the final byte of an input file is not a newline, grep silently supplies one. Since newline is also a separator for the list of patterns, there is no way to match newline characters in a text.
- The general synopsis of the grep command line is
- grep options pattern input_file_names
- There can be zero or more options. pattern will only be seen as such (and not as an input_file_name) if it wasn't already specified within options (by using the '-e pattern' or '-f file' options). There can be zero or more input_file_names.
- https://www.linode.com/docs/tools-reference/tools/how-to-grep-for-text-in-files/

# grep

- While straightforward pattern matching is sufficient for some filtering tasks, the true power of grep lies in its ability to use regular expressions for complex pattern matching. Most characters in regular expressions match with input data literally; however, there are some sequences that carry special significance:

| Symbol | Result |
|---|---|
| . | Matches any character. |
| * | Matches zero or more instances of the preceding character. |
| + | Matches one or more instances of the preceding character. |
| [] | Matches any of the characters within the brackets. |
| () | Creates a sub-expression that can be combined to make more complicated expressions. |
| \| | **OR** operator; (www\|ftp) matches either “www” or “ftp”. |
| ^ | Matches the beginning of a line. |
| $ | Matches the end of the line. |
| \ | Escapes the following character. Since . matches any character, to match a literal period you would need to use \.. |

Prepared By: Tejaas Mukunda Reddy, Shivansh Vijay Nathan

Reference Prof. David Heckathorn

# grep

- One popular use of grep is to extract useful information from system logs:

```
# grep -Eoc "^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}.* 200"  /srv/www/example.com/logs/access.log
```

- In this command, grep filters an Apache access log for all lines that begin with an IP address, followed by a number of characters, a space and then the characters 200 (where 200 represents a successful HTTP connection). The -c option outputs only a count of the number of matches. To get the output of the IP address of the visitor and the path of the requested file for successful requests, omit the -c flag:

```
# grep -Eo "^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}.* 200"  /srv/www/example.com/logs/access.log
```

- The curly brackets specify the number of instances of the pattern. {1,3} requires that the previous character occur at least once, but no more than three times. The character class [0-9] will match against one or more numeric digits. You can also generate similar output but report on unsuccessful attempts to access content:

```
# grep -Eo "^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}.* 404"  /srv/www/example.com/logs/access.log
```

# grep

- The following command generates a list of all IP addresses that have attempted to connect to your web server. Using the -o option, only the matching strings are sent to standard output. This output is filtered through the utility uniq with the pipe operator (|) to filter out duplicate entries:

```
# grep -Eo "^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}" /srv/www/example.com/logs/access.log | uniq
```

- The next example uses an alternative pattern for matching an IP address in a different log. The following command searches the most recent /var/log/auth.log file for invalid login attempts:

```
# grep -Eo "Invalid user.*([0-9]{1,3}\.){3}[0-9]{1,3}" /var/log/auth.log
```

- You can split the above command into two layers to output a list of IP addresses with failed login attempts to your system:

```
# grep "Invalid user" /var/log/auth.log | grep -Eo "([0-9]{1,3}\.){3}[0-9]{1,3}" | uniq
```

- grep can filter the output of commands such as tail -F to provide real-time monitoring of specific log events:

```
# tail ~/.procmail/procmail.log -F | grep "Subject"
```

- In this case, tail follows the ~/procmail/procmail.log file. This output is passed to grep, which filters the stream and prints only lines that contain the string “Subject”.

# ping

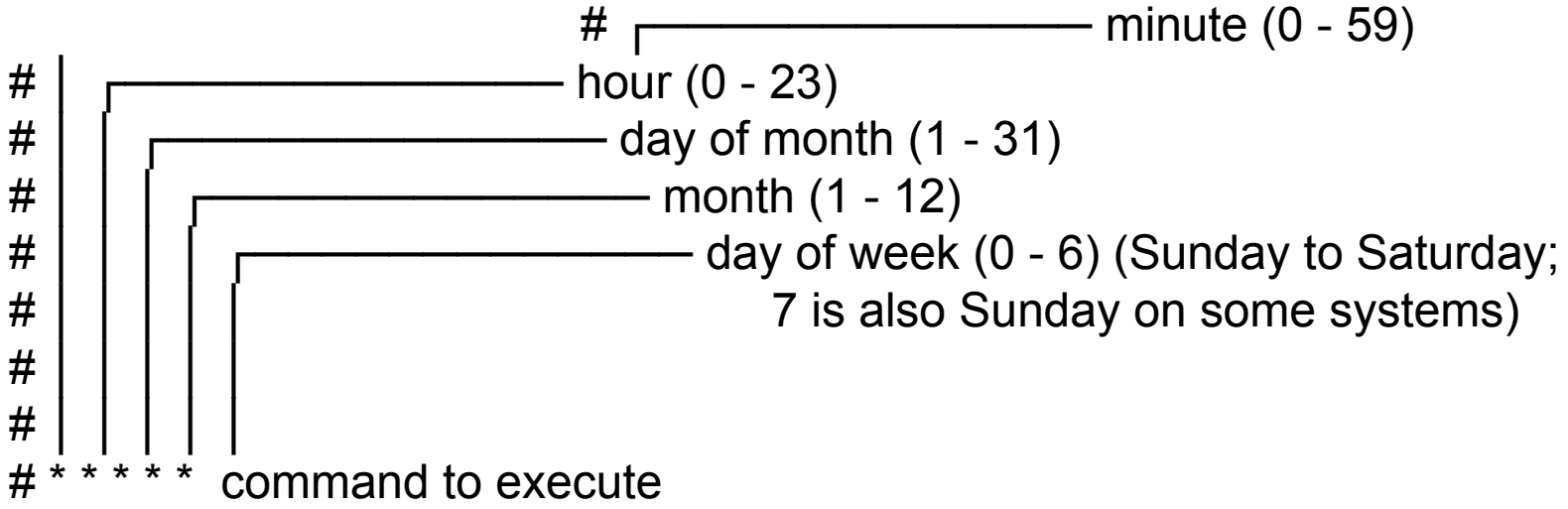
- The ping command is one of the most often used networking utilities for troubleshooting network problems.
- https://www.lifewire.com/ping-command-2618099
- The ping command is a Command Prompt command used to test the ability of the source computer to reach a specified destination computer. The ping command is usually used as a simple way to verify that a computer can communicate over the network with another computer or network device.
- The ping command operates by sending Internet Control Message Protocol (ICMP) Echo Request messages to the destination computer and waiting for a response.
- How many of those responses are returned, and how long it takes for them to return, are the two major pieces of information that the ping command provides.
- For example, you might find that there are no responses when pinging a network printer, only to find out that the printer is offline and its cable needs replaced. Or maybe you need to ping a router to verify that your computer can connect to it, to eliminate it as a possible cause for a networking issue.
- The ping command is usually used as a simple way to verify that a computer can communicate over the network with another computer or network device. The ping command operates by sending Internet Control Message Protocol (ICMP) Echo Request messages to the destination computer and waiting for a response.

Prepared By: Tejaas Mukunda Reddy, Shivansh Vijay Nathan

Reference Prof. David Heckathorn

# crontab

- https://en.wikipedia.org/wiki/Cron
- The software utility cron is a time-based job scheduler in Unix-like computer operating systems. People who set up and maintain software environments use cron to schedule jobs (commands or shell scripts) to run periodically at fixed times, dates, or intervals. It typically automates system maintenance or administration—though its general-purpose nature makes it useful for things like downloading files from the Internet and downloading email at regular intervals.
- Cron is driven by a crontab (cron table) file, a configuration file that specifies shell commands to run periodically on a given schedule. The crontab files are stored where the lists of jobs and other instructions to the cron daemon are kept. Users can have their own individual crontab files and often there is a system-wide crontab file (usually in /etc or a subdirectory of /etc) that only system administrators can edit.
- Each line of a crontab file represents a job, and looks like this:




Prepared By: Tejaas Mukunda Reddy, Shivansh Vijay Nathan

Reference Prof. David Heckathorn

# crontab

- The syntax of each line expects a cron expression made of five fields, followed by a shell command to execute.
- While normally the job is executed when the time/date specification fields all match the current time and date, there is one exception: if both "day of month" (field 3) and "day of week" (field 5) are restricted (not "*"), then one or both must match the current day.[3]
- For example, the following clears the Apache error log at one minute past midnight (00:01) every day, assuming that the default shell for the cron user is Bourne shell compliant:
- 1 0 * * * printf "" > /var/log/apache/error_log
- This example runs a shell program called export_dump.sh at 23:45 (11:45 PM) every Saturday.
- 45 23 * * 6 /home/oracle/scripts/export_dump.sh
- The configuration file for a user can be edited by calling crontab -e regardless of where the actual implementation stores this file
- Some cron implementations, such as the popular 4th BSD edition written by Paul Vixie and included in many Linux distributions, add a sixth field: an account username that runs the specified job (subject to user existence and permissions). This is allowed only in the system crontabs—not in others, which are each assigned to a single user to configure. The sixth field is alternatively sometimes used for year instead of an account username—the nncron daemon for Windows does this.
- Nonstandard predefined scheduling definitions[edit]

# crontab

- Some cron implementations[4] support the following non-standard macros:

| Entry | Description | Equivalent to |
|---|---|---|
| @yearly (or @annually) | Run once a year at midnight of 1 January | 0 0 1 1 * |
| @monthly | Run once a month at midnight of the first day of the month | 0 0 1 * * |
| @weekly | Run once a week at midnight on Sunday morning | 0 0 * * 0 |
| @daily | Run once a day at midnight | 0 0 * * * |
| @hourly | Run once an hour at the beginning of the hour | 0 * * * * |
| @reboot | Run at startup | N/A |

# crontab

- @reboot configures a job to run once when the daemon is started. Since cron is typically never restarted, this typically corresponds to the machine being booted. This behavior is enforced in some variations of cron, such as that provided in Debian,[5] so that simply restarting the daemon does not re-run @reboot jobs.
- @reboot can be useful if there is a need to start up a server or daemon under a particular user, and the user does not have access to configure init to start the program.
- cron permissions[edit]
- These two files play an important role:
- /etc/cron.allow - If this file exists, it must contain your username for you to use cron jobs.
- /etc/cron.deny - If the cron.allow file does not exist but the /etc/cron.deny file does exist then, to use cron jobs, you must not be listed in the /etc/cron.deny file.
- Note that if neither of these files exist then, depending on site-dependent configuration parameters, either only the super user can use cron jobs, or all users can use cron jobs.

# crontab

- CRON expression. A CRON expression is a string comprising five or six fields separated by white space[10] that represents a set of times, normally as a schedule to execute some routine.
- Comments begin with a comment mark #, and must be on a line by themselves.

| Field | Required | Allowed values | Allowed special characters | Remarks |
|---|---|---|---|---|
| Minutes | Yes | 0–59 | * , - | |
| Hours | Yes | 0–23 | * , - | |
| Day of month | Yes | 1–31 | * , - ? L W | ? L W only in some implementations |
| Month | Yes | 1–12 or JAN–DEC | * , - | |
| Day of week | Yes | 0–6 or SUN–SAT | * , - ? L # | ? L W only in some implementations |
| Year | No | 1970–2099 | * , - | This field is not supported in standard/default implementations. |

# crontab

- The month and weekday abbreviations are not case-sensitive.
- In the particular case of the system crontab file (/etc/crontab), a user field inserts itself before the command. It is generally set to 'root'.
- In some uses of the CRON format there is also a seconds field at the beginning of the pattern. In that case, the CRON expression is a string comprising 6 or 7 fields.[11]
- Comma ( , )
  - Commas are used to separate items of a list. For example, using "MON,WED,FRI" in the 5th field (day of week) means Mondays, Wednesdays and Fridays.
- Hyphen ( - )
  - Hyphens define ranges. For example, 2000–2010 indicates every year between 2000 and 2010, inclusive.
- Percent ( % )
  - Percent-signs (%) in the command, unless escaped with backslash (\), are changed into newline characters, and all data after the first % are sent to the command as standard input.

# mount

- https://www.computerhope.com/unix/umount.htm
- The mount command mounts a storage device or filesystem, making it accessible and attaching it to an existing directory structure.
- The umount command "unmounts" a mounted filesystem, informing the system to complete any pending read or write operations, and safely detach it.
- All files accessible in Unix, or a Unix-style system such as Linux, are arranged in one big tree: the file hierarchy, rooted at /. These files can be spread out over several devices. The mount command attaches a filesystem, located on some device or other, to the file tree. Conversely, the umount command will detach it again.
- The standard form of the mount command is: # mount -t type device dir
- This tells the kernel to attach the filesystem found on device (which is of type type) at the directory dir. The previous contents (if any), owner, and mode of dir become invisible, and as long as this filesystem remains mounted, the pathname dir refers to the root of the filesystem on device.
- The file /etc/fstab may contain lines describing what devices are usually mounted where, using which options. The command #mount -a [-t type] [-O optlist]
- causes all filesystems mentioned in fstab (of the proper type and/or having or not having the proper options) to be mounted as indicated, except for those whose line contains the noauto keyword. This command would typically be included in a boot script. Adding the -F option will make mount fork, so that the filesystems are mounted simultaneously.
- When mounting a filesystem mentioned in fstab or mtab, it suffices to give only the device, or only the mount point. The programs mount and umount maintain a list of currently mounted filesystems in the file /etc/mtab. If no arguments are given to mount, this list is printed.
- In general, fstab is no longer necessary for auto mounting.  Modern Linux systems do this like Windows does.

Prepared By:  Tejaas Mukunda Reddy, Shivansh Vijay Nathan

Reference Prof. David Heckathorn

# alias

- http://www.linfo.org/alias.html
- The alias command makes it possible to launch any command or group of commands (inclusive of any options, arguments and redirection) by entering a pre-set string (i.e., sequence of characters).
- That is, it allows a user to create simple names or abbreviations (even consisting of just a single character) for commands regardless of how complex the original commands are and then use them in the same way that ordinary commands are used.
- A command is an instruction given by a user to tell a computer to do something. Commands are generally issued by typing them in at the command line (i.e., an all-text user interface) and then pressing the ENTER key, which passes them to the shell. A shell is a program that provides the traditional, text-only user interface for a Unix-like operating systems. Its primary function is to read commands and then execute (i.e., run) them.
- The alias command is built into a number of shells including ash, bash (the default shell on most Linux systems), csh and ksh. It is one of several ways to customize the shell (another is setting environmental variables). Aliases are recognized only by the shell in which they are created, and they apply only for the user that creates them, unless that user is the root (i.e., administrative) user, which can create aliases for any user.
- Listing and Creating Aliases
- The general syntax for the alias command varies somewhat according to the shell. In the case of the bash shell it is
- alias [-p] [name="value"]
- When used with no arguments and with or without the -p option, alias provides a list of aliases that are in effect for the current user, i.e.,

# alias

- Uses for alias
  - Reducing the amount of typing that is necessary for commands or groups of commands that are long and/or tedious to type. These commands could include opening a file that is frequently used for studying or editing.
  - A second type of use for aliases is specifying the default version of a program that exists in several versions on a system or specifying default options for a command. For example, the following alias would have the ls command always list all items in a directory rather than just the non-hidden ones:
    - # alias ls="ls -a"
  - A third use of aliases is correcting common misspellings of commands. For example, if a user has a habit of accidentally typing pdw instead of pwd, the following command will create an alias so that either spelling will work:
    - # alias pdw="pwd"
  - A fourth use of aliases is increasing the safety of the system by making commands interactive. This forces the user to confirm that it is desired to perform a specific action and thereby reduces the risk from accidental or impulsive abuse of powerful commands. For example, the rmcommand, which can remove files and directories and make them virtually unrecoverable, can be made interactive as follows:
    - # alias rm="rm -i"
  - Likewise, the risks associated with the cp command, which is used to copy the contents of one file to another file, can also be reduced by making it interactive by default. If the name for the file to be written to does not exist in the specified directory (by default the current directory), it will be created, but if it already exists, its contents will be overwritten. Thus, creating the following alias will reduce the chances of an unintended overwriting:
    - # alias cp="cp -i"

# alias

- Another use of aliases is standardizing the name of a command across multiple operating systems. For example, different versions of Linux or other operating systems contain different versions of the vi text editor (i.e., vi or its clones vim, nvi, elvis, etc.), but issuing the vi command on any of these divergent systems will generally launch the particular vi clone that is resident on that system (assuming that the appropriate alias has been created). For instance, Red Hat Linux installs vim by default, but issuing the command vi launches vim because the alias alias vi="vim" is also installed by default for all users for the bash, csh and tcsh shells. Of course, the command vim can also be used, but vi is easier for most people to remember.
- For people accustomed to MS-DOS commands, the following aliases can be defined so that a Unix-like operating system appears to behave more like MS-DOS:
    - alias dir="ls"
    - alias copy="cp"
    - alias rename="mv"
    - alias md="mkdir"
    - alias rd="rmdir"
    - alias del="rm -i"
- However, some experienced users of Unix-like systems contend that this may not be a good idea and that it might just make Linux seem more confusing, rather than simpler. Instead, they advocate having Linux users become accustomed to the UNIX terminology right from the start

# ssh

- https://www.ssh.com/ssh/command/
- Practically every Unix and Linux system includes the ssh command. This command is used to start the SSH client program that enables secure connection to the SSH server on a remote machine. The ssh command is used from logging into the remote machine, transferring files between the two machines, and for executing commands on the remote machine.
- SSH COMMAND IN LINUX
- The ssh command provides a secure encrypted connection between two hosts over an insecure network. This connection can also be used for terminal access, file transfers, and for tunneling other applications. Graphical X11 applications can also be run securely over SSH from a remote location.
- OTHER SSH COMMANDS
- There are other SSH commands besides the client ssh. Each has its own page.
    - ssh-keygen - creates a key pair for public key authentication
    - ssh-copy-id - configures a public key as authorized on a server
    - ssh-agent - agent to hold private key for single sign-on
    - ssh-add - tool to add a key to the agent
    - scp - file transfer client with RCP-like command interface
    - sftp - file transfer client with FTP-like command interface
    - sshd - OpenSSH server

Prepared By: Tejaas Mukunda Reddy, Shivansh Vijay Nathan

Reference Prof. David Heckathorn

# ssh

- USING THE LINUX CLIENT
- Linux typically uses the OpenSSH client. The ssh command to log into a remote machine is very simple. To log in to a remote computer called sample.ssh.com, type the following command at a shell prompt:
  - # ssh sample.ssh.com
- If this is the first time you use ssh to connect to this remote machine, you will see a message like:
- The authenticity of host 'sample.ssh.com' cannot be established.
- DSA key fingerprint is 04:48:30:31:b0:f3:5a:9b:01:9d:b3:a7:38:e2:b1:0c.
- Are you sure you want to continue connecting (yes/no)?
- Type yes to continue. This will add the server to your list of known hosts (~/.ssh/known_hosts) as seen in the following message:
- Warning: Permanently added 'sample.ssh.com' (DSA) to the list of known hosts.
- Each server has a host key, and the above question related to verifying and saving the host key, so that next time you connect to the server, it can verify that it actually is the same server.
- Once the server connection has been established, the user is authenticated. Typically, it asks for a password. For some servers, you may be required to type in a one-time password generated by a special hardware token.
- Once authentication has been accepted, you will be at the shell prompt for the remote machine.

# ssh

- SSH CLIENT CONFIGURATION FILE
- The ssh command reads its configuration from the SSH client configuration file ~/.ssh/config. For more information, see the page on SSH client configuration file.
- CONFIGURING PUBLIC KEY AUTHENTICATION
- To configure passwordless public key authentication, you may want to create an SSH key and set up an authorized_keys file. See the pages on ssh-keygen and ssh-copy-id for more information.
- CONFIGURING PORT FORWARDING
- Command-line options can be used to set up port forwarding. Local fowarding means that a local port (at the client computer) is tunneled to an IP address and port from the server. Remote forwarding means that a remote port (at the server computer) is forwarded to a given IP address and port from the client machine. See the page on configuring port forwarding on how to configure them.
- OpenSSH also supports forwarding Unix domain sockets and IP packets from a tunnel device to establish a VPN (Virtual Private Network).

Prepared By: Tejaas Mukunda Reddy, Shivansh Vijay Nathan

Reference Prof. David Heckathorn

# ssh

- SSH COMMAND LINE OPTIONS
- Some of the most important command-line options for the OpenSSH client are:
  - -1 Use protocol version 1 only, -2 Use protocol version 2 only, -4 Use IPv4 addresses only, -6 Use IPv6 addresses only.
  - -A Enable forwarding of the authentication agent connection, -a Disable forwarding of the authentication agent connection, -C Use data compression
  - -c cipher_spec Selects the cipher specification for encrypting the session.
  - -D [bind_address:]port Dynamic application-level port forwarding. This allocates a socket to listen to port on the local side. When a connection is made to this port, the connection is forwarded over the secure channel, and the application protocol is then used to determine where to connect to from the remote machine.
  - -E log_file Append debug logs to log_file instead of standard error.
  - -F configfile Specifies a per-user configuration file. The default for the per-user configuration file is ~/.ssh/config.
  - -g Allows remote hosts to connect to local forwarded ports, -i identity_file A file from which the identity key (private key) for public key authentication is read.
  - -J [user@]host[:port] Connect to the target host by first making a ssh connection to the pjump host[(/iam/jump-host) and then establishing a TCP forwarding to the ultimate destination from there.
  - -l login_name Specifies the user to log in as on the remote machine, -p port Port to connect to on the remote host.
  - -q Quiet mode, -V Display the version number.
  - -v Verbose mode., -X Enables X11 forwarding.

# ssh

- A LITTLE HISTORY
- SSH replaced several older commands and protocols in Unix and Linux the 1990s. They include telnet, rlogin, and rsh.
- SSH runs at TCP/IP port 22. This is right between ftp and telnet, which are 20 years older. Read the story of how SSH got port 22.
- The following video summarizes how and why SSH was originally developed.
- https://www.youtube.com/watch?v=OHBdKM7s5V4

Prepared By: Tejaas Mukunda Reddy, Shivansh Vijay Nathan

Reference Prof. David Heckathorn

# iptables

- https://www.howtogeek.com/177621/the-beginners-guide-to-iptables-the-linux-firewall/
- iptables is a command-line firewall utility that uses policy chains to allow or block traffic. When a connection tries to establish itself on your system, iptables looks for a rule in its list to match it to. If it doesn't find one, it resorts to the default action.
- iptables almost always comes pre-installed on any Linux distribution. To update/install it, just retrieve the iptables package:
- sudo apt-get install iptables
- There are GUI alternatives to iptables like Firestarter, but iptables isn't really that hard once you have a few commands down. You want to be extremely careful when configuring iptables rules, particularly if you're SSH'd into a server, because one wrong command can permanently lock you out until it's manually fixed at the physical machine.
- Types of Chains
- iptables uses three different chains: input, forward, and output.
- Input – This chain is used to control the behavior for incoming connections. For example, if a user attempts to SSH into your PC/server, iptables will attempt to match the IP address and port to a rule in the input chain.
- Forward – This chain is used for incoming connections that aren't actually being delivered locally. Think of a router – data is always being sent to it but rarely actually destined for the router itself; the data is just forwarded to its target. Unless you're doing some kind of routing, NATing, or something else on your system that requires forwarding, you won't even use this chain.

# iptables

- There's one sure-fire way to check whether or not your system uses/needs the forward chain.
- # iptables -L -v
- The screenshot above is of a server that's been running for a few weeks and has no restrictions on incoming or outgoing connections. As you can see, the input chain has processed 11GB of packets and the output chain has processed 17GB. The forward chain, on the other hand, has not needed to process a single packet. This is because the server isn't doing any kind of forwarding or being used as a pass-through device.
- Output – This chain is used for outgoing connections. For example, if you try to ping howtogeek.com, iptables will check its output chain to see what the rules are regarding ping and howtogeek.com before making a decision to allow or deny the connection attempt.
- The caveat
- Even though pinging an external host seems like something that would only need to traverse the output chain, keep in mind that to return the data, the input chain will be used as well. When using iptables to lock down your system, remember that a lot of protocols will require two-way communication, so both the input and output chains will need to be configured properly. SSH is a common protocol that people forget to allow on both chains.
- History
- https://mediatemple.net/community/products/dv/204404624/using-the-history-command
- In Linux, there is a very useful command to show you all of the last commands that have been recently used. The command is simply called history, but can also be accessed by looking at your .bash_history in your home folder. By default, the history command will show you the last five hundred commands you have entered.
- This is especially useful during a live forensics investigation. passwd

Cal State Fullerton

# CPSC 254

## Software Development With Open Source Systems - Linux Servers

Instructor: Tejaas Mukunda Reddy

# What is linux server and why does your business need one?

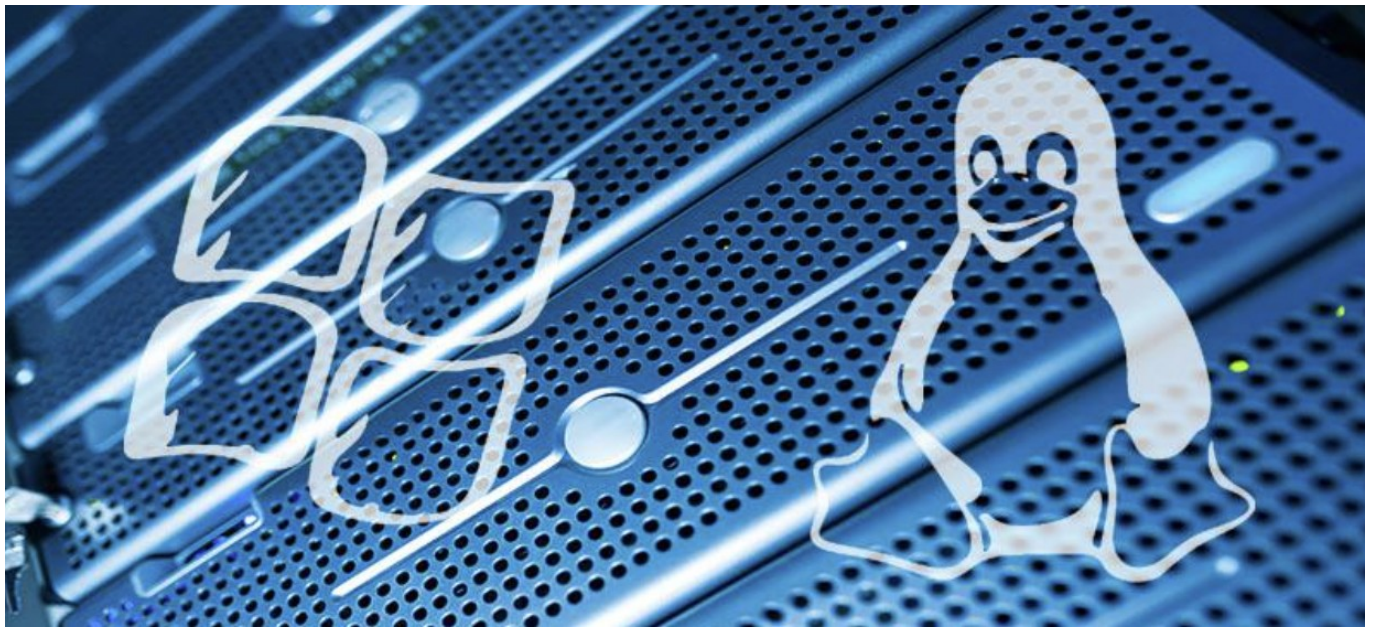
- Manage individual systems from an easy-to-use web interface that includes storage, networking, containers, services, and more.
- Automate consistency and compliance across heterogeneous multiple environments and reduce scripting rework with system roles using native configuration management tools like Ansible, Chef, Salt, Puppet, and more.
- Simplify platform updates with in-place upgrades that eliminate the hassle of machine migrations and application rebuilds.
- Resolve technical issues before they impact business operations by using predictive analytics tools to automate identification and remediation of anomalies and their root causes.
- Linux servers are powering innovation around the globe. As the platform for enterprise workloads, a Linux server should provide a stable, secure, and performance-driven foundation for the applications that run the business of today and tomorrow.
- OpenSource.com https://opensource.com/article/18/5/what-linux-server 28 May 2018 Daniel Oh (Red Hat, Correspondent) Feed

Prepared By: Tejaas Mukunda Reddy, Shivansh Vijay Nathan

Reference Prof. David Heckathorn

# Why Use Linux servers?

- Performance: Linux servers offer the performance you need your infrastructure to deliver.
- Security: Linux servers offer enhanced permissions that can be optimized for security.
- Stability: Linux servers are built on open-source technology that supports snapshot capabilities.
- Scalable: Linux works with cloud technologies to help you scale your business more easily.



Prepared By: Tejaas Mukunda Reddy, Shivansh Vijay Nathan

Reference Prof. David Heckathorn

# Linux VS Windows servers

- Linux and Microsoft Windows are the two main web-hosting services on the market. Linux is an open-source software server, which makes it cheaper and easier to use than a Windows server. Windows is a Microsoft product designed to make Microsoft a profit. For many companies, the profit is worth the price. A Windows server generally offers more range and more support than Linux servers.
- Linux is generally the choice for start-up companies while Microsoft is typically the choice of large existing companies. Companies in the middle between start-up and big companies should look to using a VPS (Virtual Private Server). Both Linux and Windows offer VPS hosting servers. A VPS runs its own duplicate of an operating system, which makes it easier for the customer to install any software that runs on the corresponding server.
- **Advantages of a Linux server or a Windows server**
- Linux open-source servers are easier to use. Some of the key benefits of a Linux server are:
- Cost savings. Open-source systems such as Linux are available to the public, which means the web hosting company only needs to pay for the technical support to install and maintain it. The technical support costs are normally spread among all the web-hosting clients, so the cost that is passed onto the client company is relatively small. With Windows, typically the company using the Windows servers either pays for the operating system or pays a periodic software license.
- Access to open-source applications. As with most technology, though, applications work better with similarly designed applications. This means a company using a Linux server should be able to seamlessly use open-source software. Using a Linux server with Windows applications is possible, but an extra layer of work to interface between the open-source technology and Windows for-profit technology will be required.
- More reliable. Linux and open-source software generally use fewer resources, making the system the more efficient.
- Easier to modify. Linux servers and software can be modified on the fly. Modifications to Windows products generally require waiting for a new version of the server to be released.

Prepared By: Tejaas Mukunda Reddy, Shivansh Vijay Nathan

Reference Prof. David Heckathorn

Cal State Fullerton

# CPSC 254

## Software Development With Open Source Systems - Linux Security

Instructor: Tejaas Mukunda Reddy

# Top 10 Best Methods on how to improve Linux security


- This blog post is created in order to help you significantly improve Ubuntu-based Linux security and to avoid the general bruteforcing, phishing as well as other types of attacks that may be targeted towards your desktop.
- According to recent statistics at Berkeley Linux Users group, the market share of Linux usage has been growing, even though not at an amazing rate. One possible reason for that is the security issues which many enterprises experienced in recent years. Besides the number of targeted attacks with various forms of banking malware, worms as well as spyware, many ransomware infections like WannaCry and EternalPetya have proven that even often-used and often-patched operating systems such as Windows 10 can be affected big time. This has driven many personal as well as enterprise users to seek alternative methods and solutions, turning their heads to Linux-based operating system. The biggest share of those have the Ubuntu-based Linux OS's. If you are a beginner Linux user and are looking for the methods to improve your security, we recommend implementing the below-suggested ones to turn your Linux distribution into a software fortress.

Prepared By: Tejaas Mukunda Reddy, Shivansh Vijay Nathan

Reference Prof. David Heckathorn

# Top 10 Best Security tools and methods for Linux

- To best provide you with the methods we have decided to divide them chronologically. This basically means that the methods that we have explained first are more important and essential for your security. Let's begin with the necessary command after installation, which is the necessary:
  - $sudo apt-get update && time
  - $sudo apt-get dist-upgrade
- In addition to this, we also do not include services that can be set up separately, like **VPN**, which is essential for Linux security and if you aim for protection you should be familiar with VPN services for Linux distros and the methods on how to set them up.
- After having those essential elements of security, you can proceed with doing other modifications on your Linux distribution.

**Encrypt your drive (full disk encryption)**

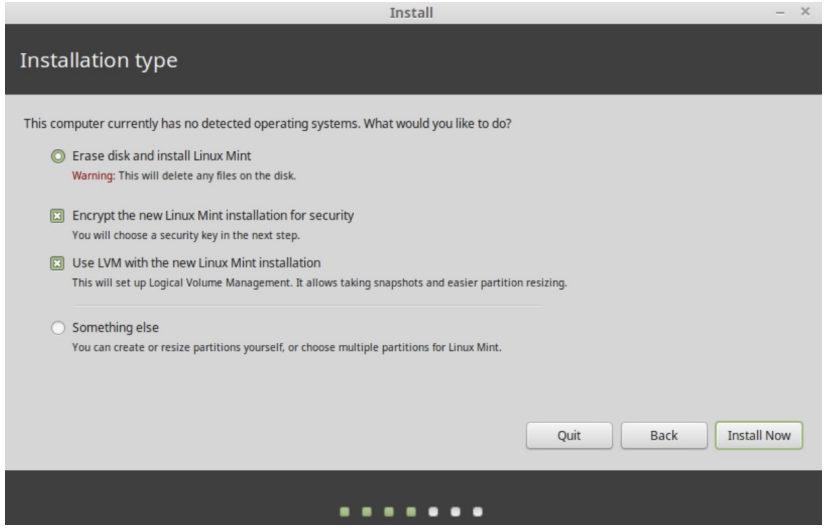
When you install your new Linux distribution, whether it is Gnome, Kubuntu, BackTrack or any other type, it's newer versions will always ask you If you would like to encrypt your drive. Whether it is an SSD drive or Hard Drive, if it is encrypted, your data remains almost 100% safe. Yes, you will be asked a password to decrypt your drive upon login, but this is the price of having a drive that only you can access to if you are familiar with your password that is. This is especially useful when you are mobile and working on a laptop.

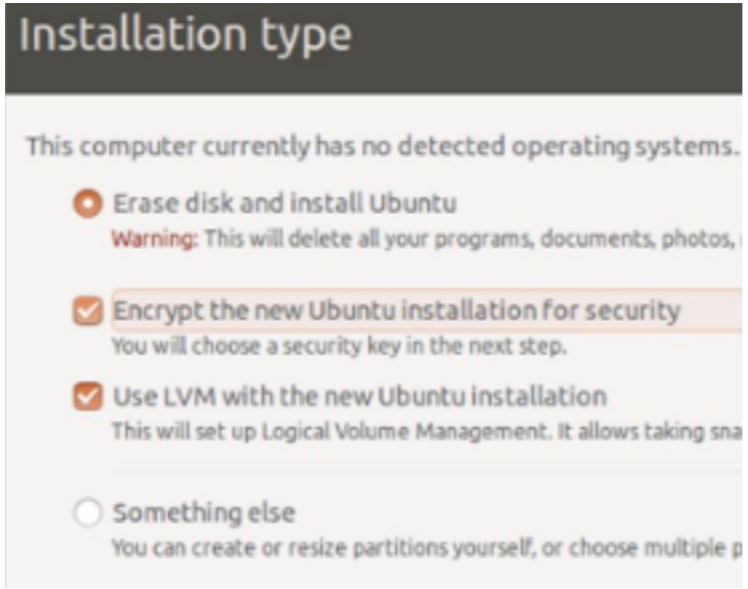
Prepared By: Tejaas Mukunda Reddy, Shivansh Vijay Nathan

Reference Prof. David Heckathorn

# Top 10 Best Security tools and methods for Linux

- If your drive is encoded and the laptop gets stolen, there is likely nothing that the criminals can do to even access your important data. Furthermore, if a hacker has physical access to your laptop and knows your account password, for example, the code required to decrypt the drive will obstruct him into entering your drive. The same principle goes if someone stole your hard drive. In addition to full disk encryption, Linux also offers to encrypt your Home folder, meaning that even if someone has access to your computer, they will not be able to tamper with everything within this crucial for Linux directory.




Prepared By: Tejaas Mukunda Reddy, Shivansh Vijay Nathan

Reference Prof. David Heckathorn

# Top 10 Best Security tools and methods for Linux

**Enable Your Firewall**

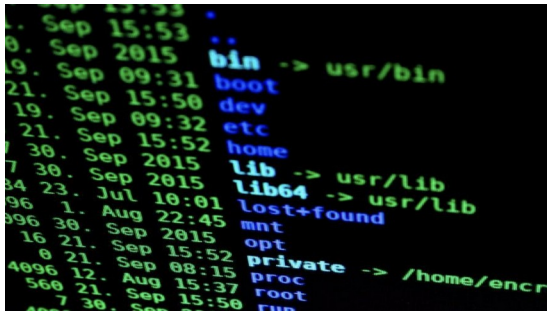


- Basically, this is one thing that every self-respecting Linux user should do when they install a Linux distribution. It more of a security ethic advise, primarily because, even with the firewall disabled, Linux has all the ports locked down either way. But, you never know, if your computer will be targeted sooner or later, because someone with hardened security obviously has something to hide and people quickly realize this. To enable your Linux firewall, you must run the Linux Terminal after which type: $sudo apt-get install guf
- GUFW stands for “Graphical Uncomplicated Fire Wall”. The command will install it and after it has finished doing so, you should open it, by typing in your terminal it’s abbreviation and hitting Enter: $gufw
- After you open GUFW you will see it’s simple user interface. From there simply click on the slider button next to Status to turn it from OFF to ON:

Prepared By: Tejaas Mukunda Reddy, Shivansh Vijay Nathan

Reference Prof. David Heckathorn

# Top 10 Best Security tools and methods for Linux




**Disable SSH Login via Root**

- In order to disable anyone logging in via SSH, you should access the file that is responsible for the configuration of SSH. The file has the following location:
  - → /etc/ssh/sshd_config
- After you have found this file, open it with the text editor app and find the following line of code and when you do, remove the # symbol from it:
  - →#PermitRootLogin no

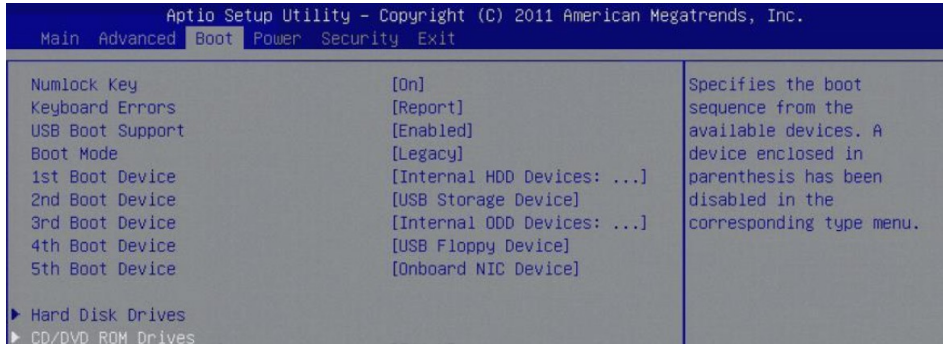
**Make your BIOS More Secure**

- This tip may not be directly Linux related but it is considered as a general security risk for most Linux distributions. After installing Linux on your computer, it is a good idea to disable any possibility of your computer to boot via USB, CD/DVD or other external drives. This means that nobody can overwrite your Linux and hence damage it or even try to access your drive by booting a Live OS. And this is just the tip of the iceberg of security threats when external boot is enabled. This is why, you should access BIOS on your system start up and go to the Boot tab from which disable the booting option from external drives:

Prepared By: Tejaas Mukunda Reddy, Shivansh Vijay Nathan

Reference Prof. David Heckathorn

# Top 10 Best Security tools and methods for Linux




In addition to these measures, add a BIOS password, which will stop someone with a physical access to your computer to enter the BIOS:

- IMPORTANT TIP! Make sure to find a way to memorize all the passwords, especially the firmware BIOS one, because there is no way of recovering it!.

**Disable USB Mount**

- One crucial method by which you can ensure higher security, especially against someone who can physically tamper with your computer is to ban them from using USB to attack it. There are many sophisticated USB-based malware which is activated automatically when the pen drive is inserted in your USB port, so this is a crucial tip to strengthen your Linux security. The only price you have to pay is to quit using USB drives all the time and find another method to safely transfer data. Here is how to do it:
- Step 1: Open any text editor and write:
  - → install usb-storage /bin/true
- Step 2: Save the file as a .conf type of file and save in the following location:
  - → /etc/modprobe.d/
- Step 3: Restart your computer and test if you are able to mount a USB drive

Prepared By: Tejaas Mukunda Reddy, Shivansh Vijay Nathan

Reference Prof. David Heckathorn

# Top 10 Best Security tools and methods for Linux

**Use Firejail Sandboxing When You Try New Applications**

- In general, Linux operating systems are designed in order to be secure by default. But this does not mean that your online browsing is not exposed against any sniffing or phishing attacks – the main reason why you need to secure yourself against new browser extensions or apps that may be unwanted on your Linux machine. Firejail is one security app that is very simple to set up and works on the latest Linux distros. Here is how to set it up on 16.04 LTS Ubuntu:
  - → sudo apt-get update
  - sudo apt-get install firejail
  - ls /etc/firejail
- Now you have successfully entered a page where you should see the profiles of all the programs installed on your computer. They should look somewhat like the following:
  - → skype.profile
  - dropbox.profile
  - icedove.profile
  - Tor.profile
- If we would like to secure Tor web browser, for example, we can use the “firejail” command in the following syntax:
  - → firejail firefox
- After entering this command, the next time you run Tor browser, the events that are happening are recorded on a log file. Here is more information on how to tinker with Firejail, kindly provided by OSTechNix.

Prepared By: Tejaas Mukunda Reddy, Shivansh Vijay Nathan

Reference Prof. David Heckathorn

# Top 10 Best Security tools and methods for Linux

**Use Anti-Virus**

- “Anti-Virus for Linux?” you would ask yourself in a critical manner… While some consider anti-virus to be completely unnecessary for a Linux, because most malware is generally designed for Windows and will not be activated on your system. This is true, however there are also arguments that there are sophisticated attacks and malware for Linux OS’s out there and they are increasing and you have no way of defending yourself once you get attacked by them, even if the probability is low. You can never be safe enough, right? There are many Linux anti-virus programs out there, and you can use Google to find the ones which are suitable most for your situation. Furthermore, we will soon post a comparison between the best anti-virus programs for Linux-based OS’s, so we suggest that you follow our blog regularly. In the meantime, you can go ahead an check MakeUseOf’s post.

# Top 10 Best Security tools and methods for Linux

**Enable Root Mode and Secure it**

- The root access is essential to administrative control of your Linux operating system and it's enabling it crucial for it too. To enable logins via root, you should use the following commands:
  - → sudo passwd root
- After this has been done,, you should type your password. Then, type the following command:
  - → sudo passwd -u root
- In order to enter the root mode, simply type:
  - → sudo -i
- When you type it, you will be requested to enter your password and you will be in privileged root mode afterwards.

# Top 10 Best Security tools and methods for Linux

**Improve Your Browsing Security**

- The most disregarded aspect of online security is the web browser, since it is the most often method by which malware may slither onto your computer system. JavaScripts, drive-by downloads, worms and many other types of malware may find their way onto your Linux system.
- There are multiple ways to improve browser security. For starters, you can add browser extensions(Add-ons) which can contribute to the improvement of security and preventing attacks from suspicious third-parties. Here we have some suggestions on such:
- Adblock Plus – an add-blocker which can disable any pop-ups, redirects and other types of advertisements. Available for most web browsers.
- Disconnect – A browser extension which aims to make your web browsing private, since it aims to prevent third-parties to track your browsing history.
- Privacy Badger – Created by the notorious EFF (Electronic Frontier Foundation), this browser add-on helps to block malicious adverts as well as tracking technologies and categorizes the danger of every site, based on colors (green, yellow, red).
- NoScript – Possibly the most useful browser add-on for Firefox. It aims to block absolutely any script that is executed from websites you do not trust. This basically means that you can use the script to white-list multiple sites you visit often and trust and when it comes to the new websites, they will be blocked. Significantly increases security.

Prepared By: Tejaas Mukunda Reddy, Shivansh Vijay Nathan

Reference Prof. David Heckathorn

# Top 10 Best Security tools and methods for Linux

**Improve Your Browsing Security**

- However, if you still feel that your web browser is not secure enough, because you have read about bug bounties and other details that decrease your trust in your browser, you can try and download web browsers which are specifically oriented towards security. This is why we have the following suggestions which you can try and choose the best security browser for Linux that will suit your activity:
  - Elinks
  - Konqueror
  - Midori
  - Opera
  - Slimboat

Prepared By: Tejaas Mukunda Reddy, Shivansh Vijay Nathan

Reference Prof. David Heckathorn