https://www.youtube.com/watch?v=2C7mNr5WMjA

CSI3660 – System Administration

Prof. Fredericks

Logging and Packages

Outline

- Logging
- Installing software
  - Compiling and installing
  - Package management

Logging

- Log file
  - File containing system or debugging information
  - Typically recorded during daemon activity
  - Information includes error messages
  - Text file, SQL database, other workstations, etc.
- What gets logged?
  - Kernel information
  - Daemon activity
  - Application data
- Log files are usually where you look **first** when troubleshooting

But first…

- Let’s revisit yet another useful command: grep!
- Why?
  - Searching:
    - Logfiles for information
    - Output of commands for specific lines
    - etc

grep

- grep <keywords>
- grep “key words”

```
$ grep hello world file1
```

- Searches for **hello** in a file called world and a file called file1

```
$ grep "hello world" file1
```

- Searches for **hello world** in a file called file1

```
$ grep -n hello file1
```

- Prints the line number that **hello** is found in file1
- -i would make it case-insensitive (grep –in hello file1)

grep

- How you'll probably use it:

$ cat <file> | grep <search terms>

  - Run some command that generates a lot of output (cat <file>)
  - Search that output (pipe it in) to grep

- E.g.,

```
~~$ chkconfig | grep httpd~~
```

```
$ systemctl list-unit-files | grep cups
```

Other Handy Commands

- cat
  - Dump all information from file to command line
- tail
  - View last 10 lines of file (generally the ‘newest’)
- head
  - View first 10 lines of file (generally the ‘oldest’)

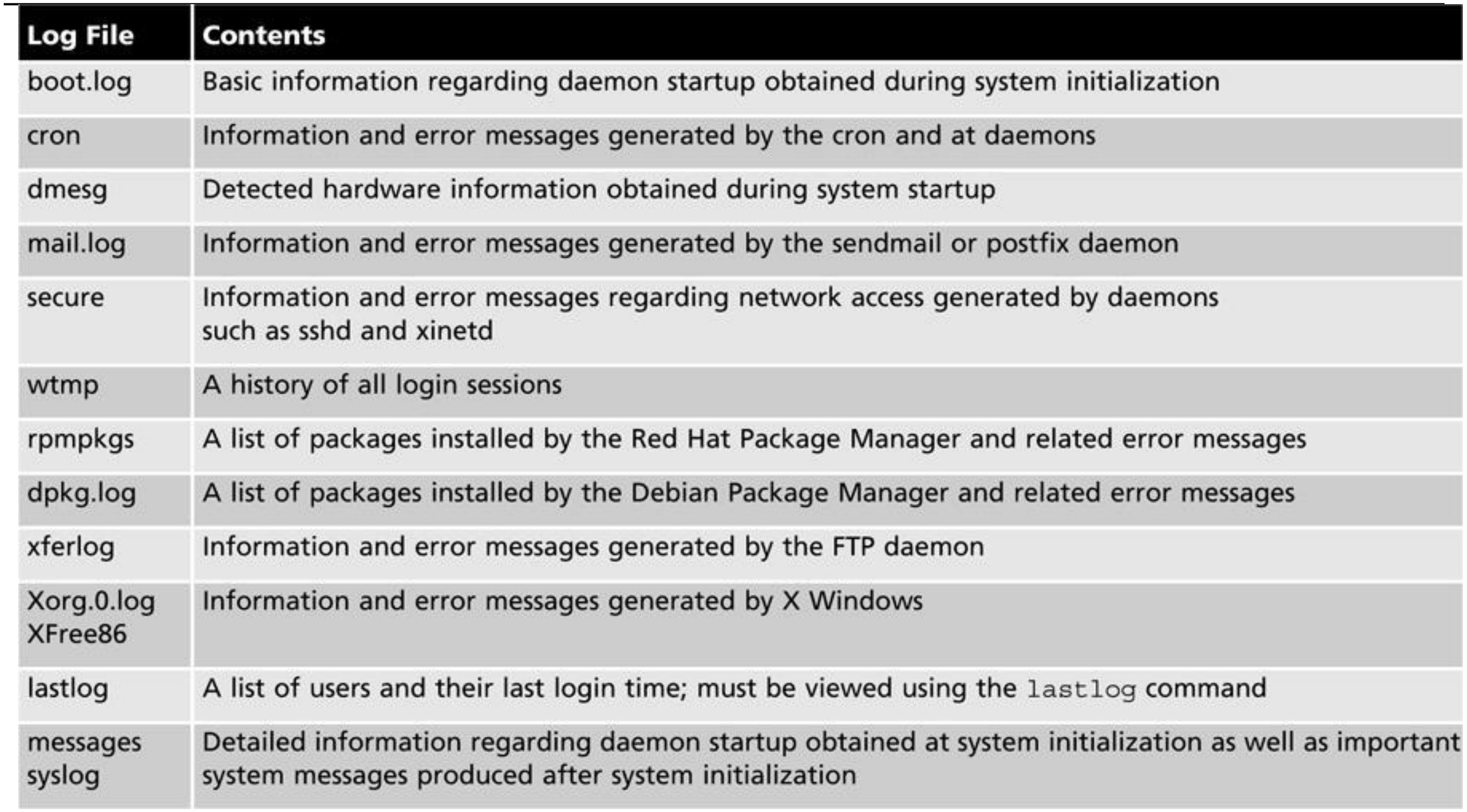
Logging

- /var/log/
  - Contains most log files
  - Many programs store their log files in subdirectories of the /var/log directory

- The two most common logging daemons used on Linux systems today are:
  - System Log Daemon (rsyslogd)
  - Systemd Journal Daemon (journald)
  - **JournalCTL → new logging method**

- Can also log by simply printing debugging/error statements

Log File Administration



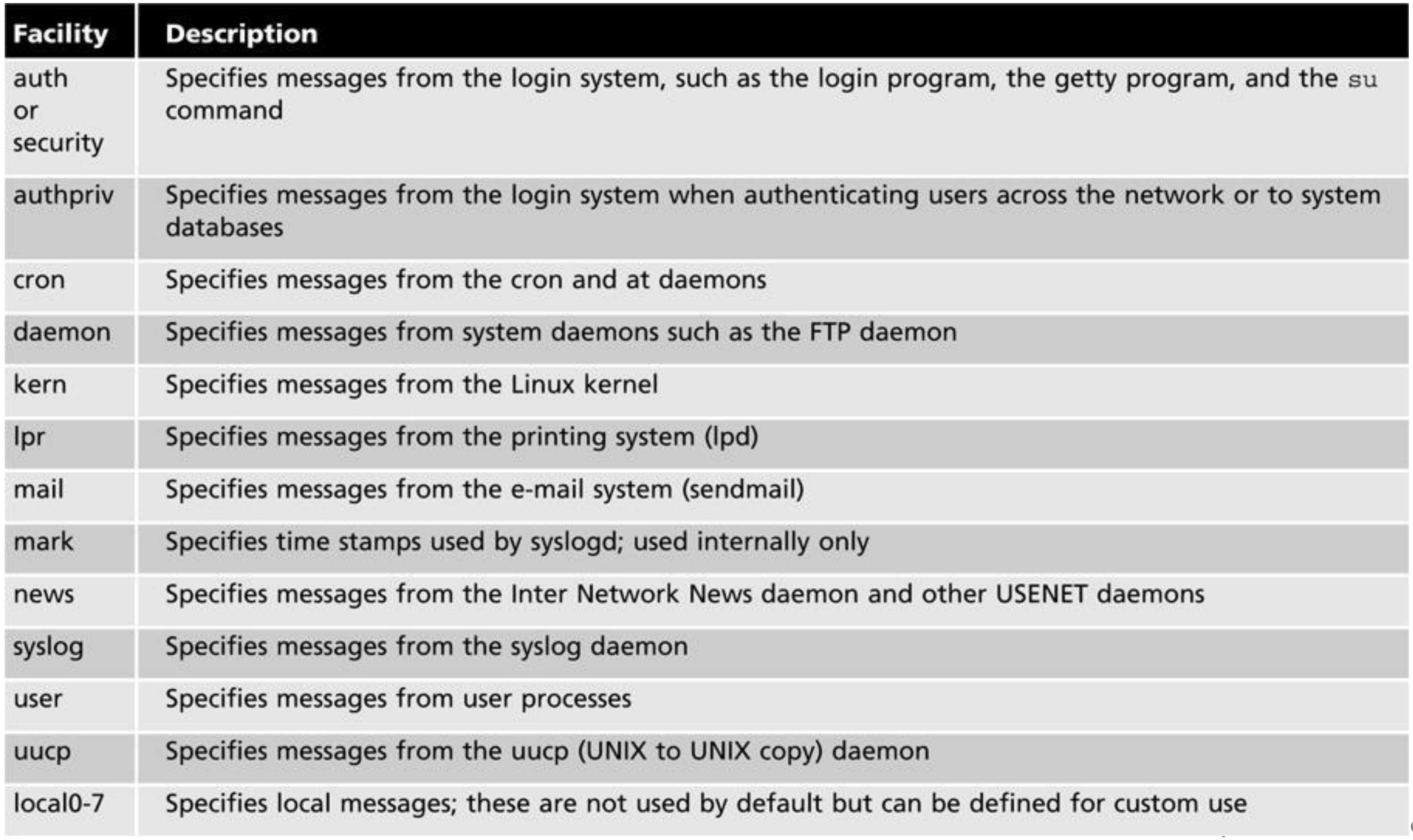
Log File	Contents
boot.log	Basic information regarding daemon startup obtained during system initialization
cron	Information and error messages generated by the cron and at daemons
dmesg	Detected hardware information obtained during system startup
mail.log	Information and error messages generated by the sendmail or postfix daemon
secure	Information and error messages regarding network access generated by daemons such as sshd and xinetd
wtmp	A history of all login sessions
rpmpkgs	A list of packages installed by the Red Hat Package Manager and related error messages
dpkg.log	A list of packages installed by the Debian Package Manager and related error messages
xferlog	Information and error messages generated by the FTP daemon
Xorg.0.log XFree86	Information and error messages generated by X Windows
lastlog	A list of users and their last login time; must be viewed using the `lastlog` command
messages syslog	Detailed information regarding daemon startup obtained at system initialization as well as important system messages produced after system initialization

Table 10-3: Common Linux log files found in /var/log

Working with the System Log Daemon

- System log daemon (syslogd)
  - The traditional and most common logging daemon
  - Creates /dev/log socket for system processes to write to
  - Writes to appropriate log file using /etc/rsyslog.conf file
    - Entries indicate facility and priority
- Facility
  - Area of system that information is gathered from
  - "Program"
- Priority
  - Importance of system information

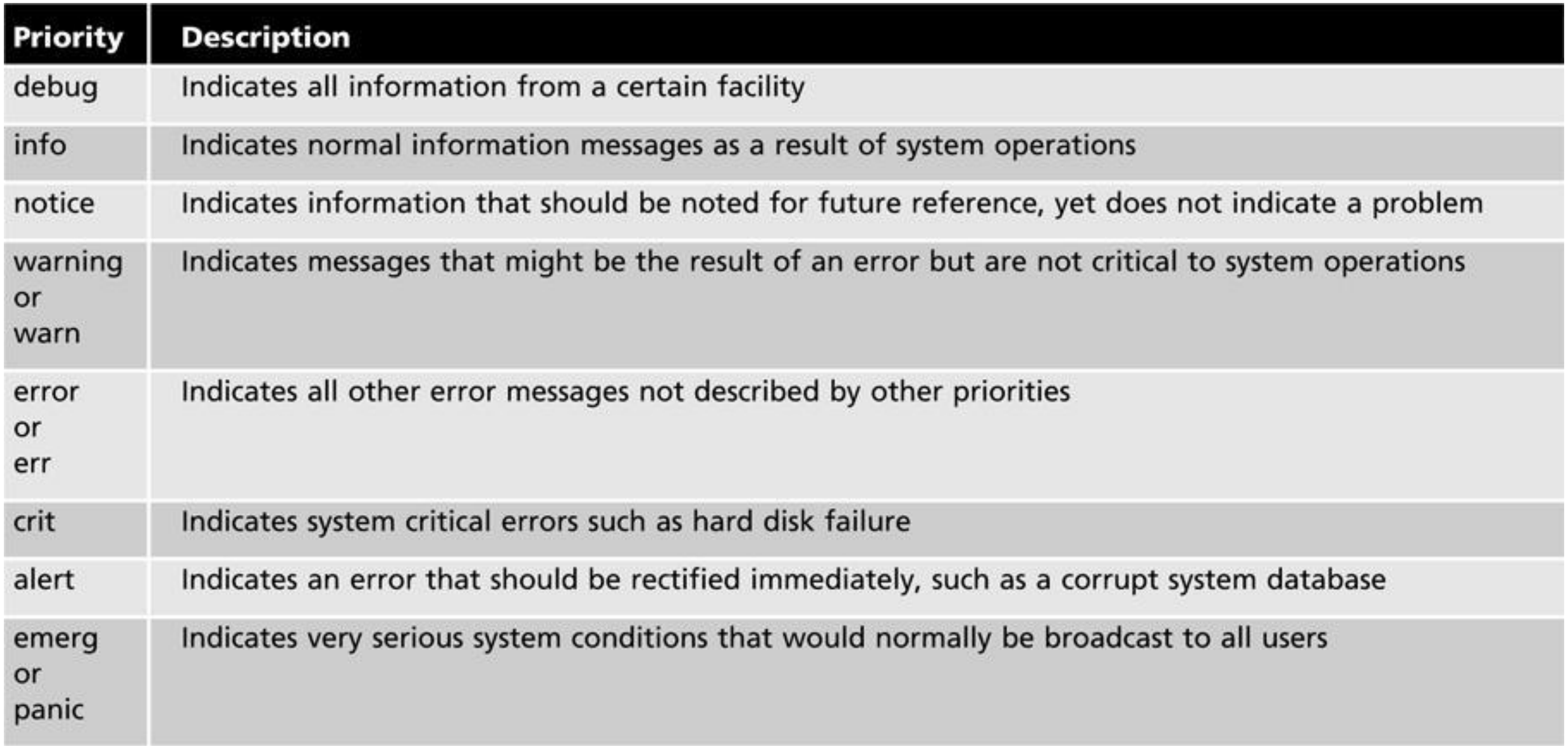
Working with the System Log Daemon



Facility	Description
auth or security	Specifies messages from the login system, such as the login program, the getty program, and the `su` command
authpriv	Specifies messages from the login system when authenticating users across the network or to system databases
cron	Specifies messages from the cron and at daemons
daemon	Specifies messages from system daemons such as the FTP daemon
kern	Specifies messages from the Linux kernel
lpr	Specifies messages from the printing system (lpd)
mail	Specifies messages from the e-mail system (sendmail)
mark	Specifies time stamps used by syslogd; used internally only
news	Specifies messages from the Inter Network News daemon and other USENET daemons
syslog	Specifies messages from the syslog daemon
user	Specifies messages from user processes
uucp	Specifies messages from the uucp (UNIX to UNIX copy) daemon
local0-7	Specifies local messages; these are not used by default but can be defined for custom use

Table 10-4: Facilities used by the System Log Daemon

Working with the System Log Daemon



Priority	Description
debug	Indicates all information from a certain facility
info	Indicates normal information messages as a result of system operations
notice	Indicates information that should be noted for future reference, yet does not indicate a problem
warning or warn	Indicates messages that might be the result of an error but are not critical to system operations
error or err	Indicates all other error messages not described by other priorities
crit	Indicates system critical errors such as hard disk failure
alert	Indicates an error that should be rectified immediately, such as a corrupt system database
emerg or panic	Indicates very serious system conditions that would normally be broadcast to all users

Table 10-5: Priorities used by the System Log Daemon

Facility/Priority Selectors

- Configuration file formatting:

  - facility.level   action
  - facility1.facility2.level   action
  - facility1.level1;facility2.level2   action
  - *.level   action

Standalone Machine Example

```
# Simple standalone machine / small network

# emergencies – tell all logged in

.emerg                *

# important messages

*.warning;daemon,auth.info;user.none   /var/log/messages

# printer errors

lpr.debug                /var/log/pld-errs
```

Differences between syslogd and rsyslogd

- rsyslogd is based on syslogd
  - Can use a standard syslogd configuration file
  - Enhanced features
    - Listen to TCP/UDP (network packets)
    - Advanced log filtering
      - E.g., send output from particular PID to file
    - Buffered logging

Working with the Systemd Journal Daemon

- The Systemd Journal Daemon
  - Replaces the System Log Daemon on Linux distributions that use Systemd
    - Such as Fedora 20
- Similar to System Log Daemon
  - Events logged are not controlled by specific rules
- Journald logs all information to a database under the /var/log/journal directory structure
  - Events are tagged with the same facility and priority information as the rsyslogd daemon

Working with the Systemd Journal Daemon

- **`journalctl`**
  - View events within the journald database
- Type **`journalctl`** and press the Tab key to see a list of areas and criteria that can be queried
- Query events related to a specific process or daemon
  - If you specify the path name to the executable file or PID
    - $ journalctl _UID=1000

Logging Practices

- Each corporation has their own specific needs
- Things to consider:
  - Available space on server/workstation
  - When users are performing tasks
  - What actually is necessary to track?

Logging Best Practices

- Keep track of **critical** failures

- Don't "overproduce"
  - Only log what is necessary
  - Ensure your team/yourself doesn't get buried in log output
  - Watch the most important information as necessary

- Separate debug and access logs
  - Access logs show who/what were accessing resource
  - Debug logs show behavior of program/command
  - Easier to parse when they are separated

- Cleanup old logs!

Basic Necessities for Logging

- Consider logging:
  - Username / user ID
  - Success / failure of event
  - If network, source address
  - Date and time FROM RELIABLE SOURCE – NTP
  - If sensitive data was updated/removed/added
  - Necessary details of event

Log Storage

- Logs should:
  - Be placed on their own partition
  - Be locally accessible for a predetermined amount of time
    - For example, 3 months
  - Be archived to long term storage for predetermined amount of time
    - For example, 1 year
  - Be purged if unnecessary!

- But, access should be limited!

Example

- Let’s do some logging with rsyslogd
- Send all info messages to a special log
  - (Edit /etc/rsyslog.conf)

```
# Log all messages to messages file
*.*    /var/log/emf-messages
```

  - (sudo systemctl restart rsyslog.service)
  - (check /var/log/emf-messages)

More Examples!

- Log to system log file

```
[user]$ logger "This is a test log statement"
```

- Check /var/log/messages

- logger:
  - Sends a manual log entry
  - Default priority/facility is **user.notice**
  - Good for logging your own scripts!

Something Practical

- Let’s create a script to add users to our system
- BUT, we need to log this behavior

Package Management

Software Installation

- Software for Linux can be:
  - Binary files precompiled to run on certain hardware architectures
  - Source code, which must be compiled before use
- Package manager
  - System that defines standard package format
  - Used to install, query, and remove packages
- Red Hat Package Manager (RPM)
  - Common package manager used by Linux systems today (including Scientific)
  - Debian-based Linux distributions, such as Ubuntu, use the Debian Package Manager (DPM)
  - ArchLinux uses pacman

Compiling Source to Programs

- Procedure for compiling source code into binary programs
  - (Mostly) standardized among most OSS developers
- **`./configure`**
  - Configures makefile
  - Checks dependencies
  - Handles user preferences

Compiling Source to Programs

- **`make`**
  - Looks for the Makefile and uses it to compile the source code into binary using the appropriate compiler
  - Makefile
    - Contains most of the information and commands necessary to compile a program
      - Libraries, compilation commands, etc.
- **`make install`**
  - Copies complied executable programs to designated location
  - Can typically install to system or to local (i.e., only accessible to you)

Compiling Source Code into Programs

- Uninstall
  - Depends on program
  - Makefile may have provided uninstall option
    - **`make uninstall`**

  - Or, manually delete installed fines/binaries

Example Compilation Prerequisites

- Need dependencies:
  - Compiler tools, encoding libraries, etc.
  - Need RPMFusion and EPEL repositories

```
sudo yum install epel-release   # if not included already
sudo yum group install "Development Tools"
```

# list repositories

```
yum repolist
```

# Required packages

```
sudo yum install yasm libXext-devel libXfixes-devel zlib-devel
```

```
# just in case
yum-config-manager --add-repo http://www.nasm.us/nasm.repo
yum install autoconf automake bzip2 cmake freetype-devel gcc gcc-c++
git libtool make mercurial nasm pkgconfig zlib-devel
```

Example Compilation

- Let's install ffmpeg – a multimedia encoder utility
- First, get file:

```
wget http://ffmpeg.org/releases/ffmpeg-2.8.tar.bz2
```

- Unzip:

```
tar -xjvf ffmpeg-2.8.tar.gz2
```

Example Compilation

- Configure, make, and make install

```
./configure --enable-shared --enable-nonfree --enable-gpl
--enable-decoder=aac --enable-demuxer=mov --enable-x11grab
--enable-zlib --enable-protocol=http --enable-filter=aformat
--enable-filter=volume --enable-filter=aresample
       (these flags are specific to ffmpeg)

make
   (already done…takes quite a while for this program)

sudo make install
```

- If I wanted to install only for me, I would have passed --`prefix=$HOME` to install to my home directory

Run ffmpeg

- Oops!

```
[user]$ ffmpeg

ffmpeg: error while loading shared libraries:
libavdevice.so.56: cannot open shared object file: No such
file or directory
```

- https://forum.ivorde.com/ffmpeg-error-while-loading-shared-libraries-libavdevice-so-52-cannot-open-shared-object-file-no-t129.html

```
[user]$ sudo vim /etc/ld.so.conf

    Add /usr/local/lib

[user]$ sudo ldconfig                   → dynamic library linker

[user]$ ffmpeg
```

Google!

Package Managers

- Easy to manage (install, upgrade, remove)
- Central database tracks all installed packages on a system
- yum (or similar tools) handles dependency resolution
  - Yellowdog Updater Modified
  - yum installed our encoders, zlib, etc.
- Package repositories make it easy to find/install common software
  - Already added extra repositories earlier

RPM vs YUM

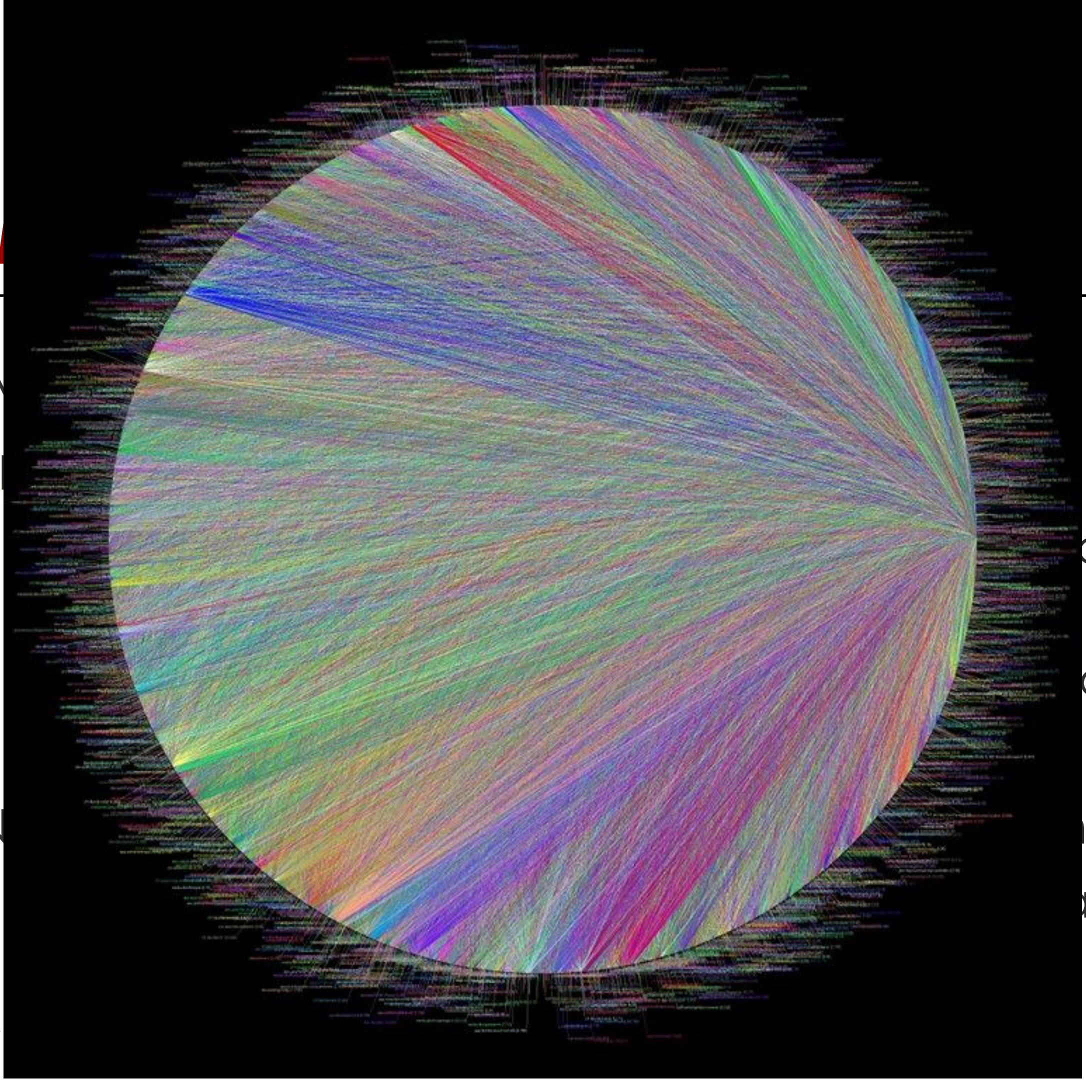
- RPM → package manager
  - “Dependency hell”
    - Can easily install packages…but…dependencies may be missing, use the wrong version, etc.
      - Package 1 → Package 2 → Package 3 → Package *n*

  - Need to install ALL dependencies for a program to run successfully
    - RPM checks for dependencies…but doesn’t actually install them

  - Yum takes care of dependency problems
    - Think of it as a frontend for RPM
    - “Metapackage manager”

RP

- RP
  - “
    - cies may be missing,
    - ackage *n*
  - N to run successfully
    - actually install them
  - Y
    - Think of it as a frontend for RPM



Another Major Difference…

- RPM
  - Must download .rpm file to install

- YUM
  - Uses repository to install!

Repositories

- Most RPM packages are located on Internet Servers
  - Called software repositories

- Add repository (software not in defaults)

```
sudo yum install epel-release    # if not included already
```

Package Versions

- EVR
  - **E**poch
  - **V**ersion
  - **R**elease

- Filename doesn’t include epoch
- foo-1.0-3 (Name: foo; Version: 1.0; Release: 3)
- foo-1.0-4 is newer
- foo-1.5-1 is even newer

Epoch

- Epoch
  - Version number ordering
  - Mainly used when RPM can’t decipher version number
  - Epoch: 42
    - Tells us that has a version number higher than epoch 41.
    - Can also be used for dependencies
      - (e.g., package requires SW Epoch 42, or >= 42:1.0)

yum info libvorbis

```
Name        : libvorbis
Arch        : x86_64
Epoch       : 1
Version     : 1.2.3
Release     : 4.el6_2.1
Size        : 2.1 M
Repo        : installed
From repo   : anaconda-ScientificLinux-201508171352.x86_64
Summary     : The Vorbis General Audio Compression Codec.
URL         : http://www.xiph.org/
License     : BSD
Description : Ogg Vorbis is a fully open, non-proprietary, patent- and
royalty-free,
            : general-purpose compressed audio format for audio and music at
fixed
            : and variable bitrates from 16 to 128 kbps/channel.
            :
            : The libvorbis package contains runtime libraries for use in programs
            : that support Ogg Vorbis.
```

Working with RPM

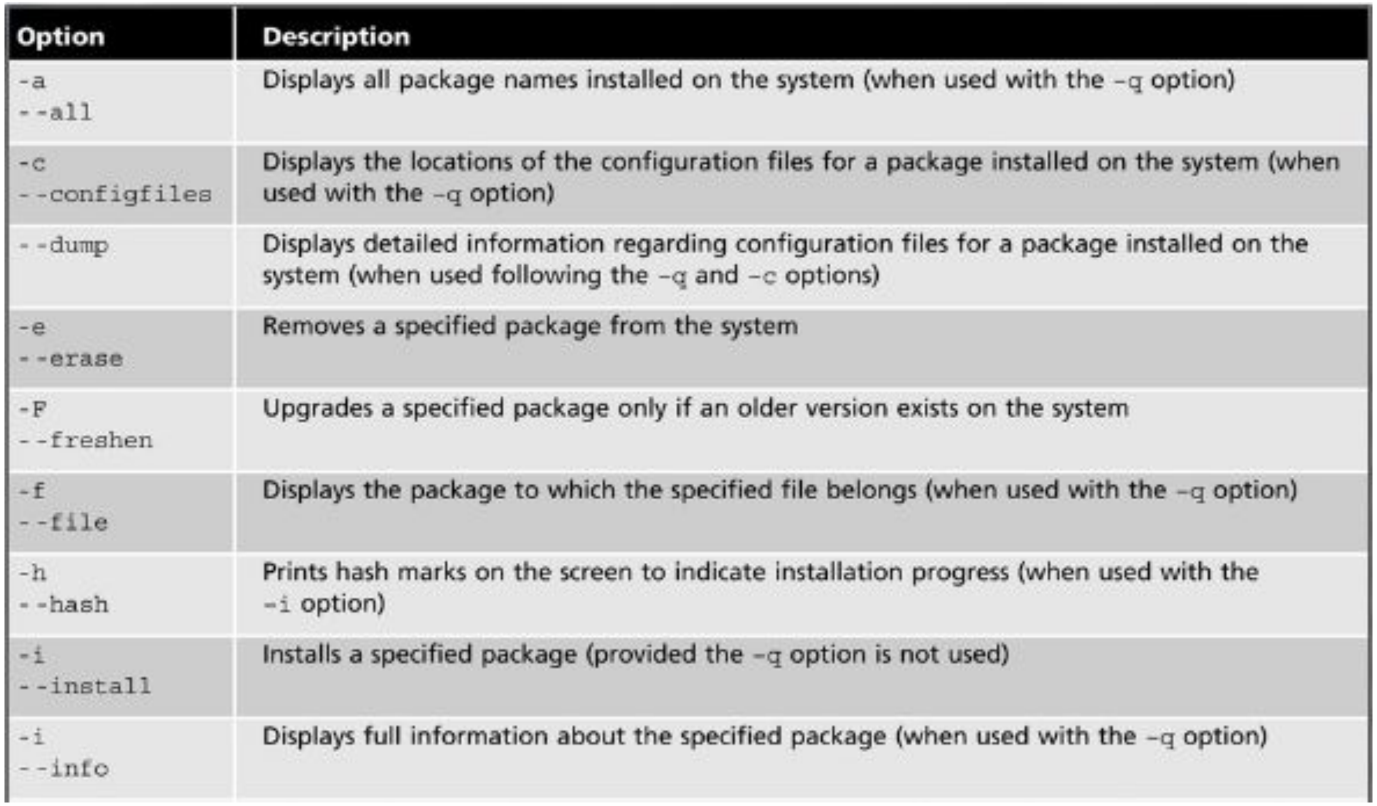
- Package dependency
  - Some RPM packages require that other RPM packages be installed on your system first
  - Error message that indicates the RPM package that needs to be installed first
  - After installing the prerequisite packages, you can successfully install your desired RPM package
  - yum takes care of this

Working with the Red Hat Package Manager (RPM)

- RPM packages filenames indicate hardware architecture for which the software was compiled
  - End with .rpm extension

- To install an RPM package, use `-i` option to `rpm` command
  - Command used to install, query, and remove RPM packages

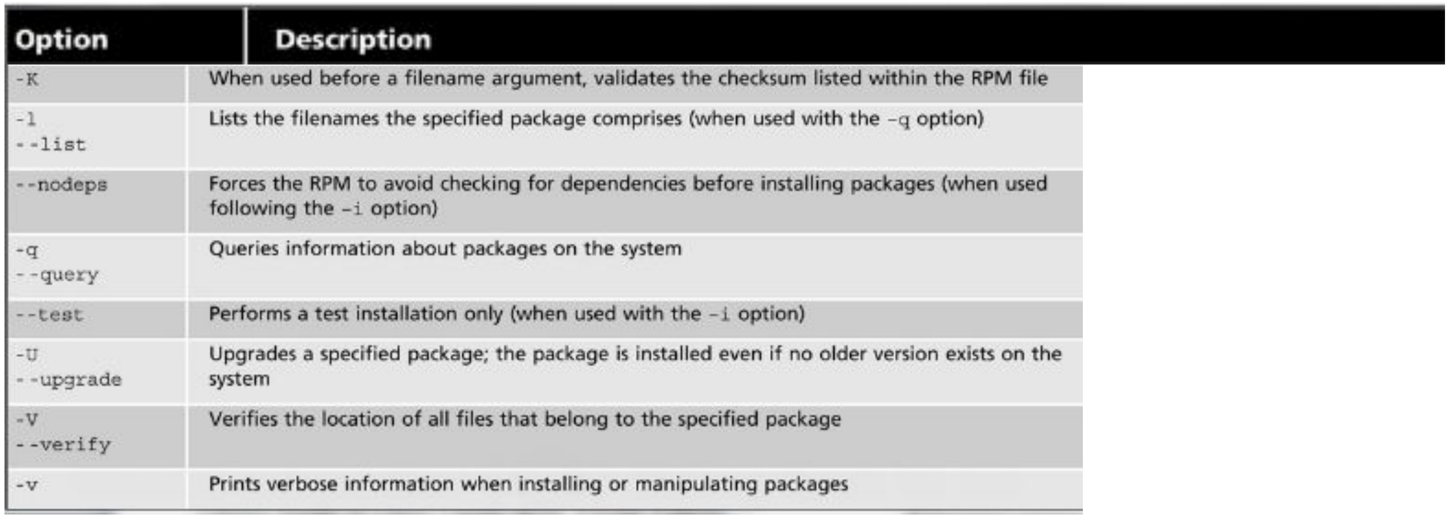
Working with the Red Hat Package Manager (RPM)



Option	Description
`-a` `--all`	Displays all package names installed on the system (when used with the `-q` option)
`-c` `--configfiles`	Displays the locations of the configuration files for a package installed on the system (when used with the `-q` option)
`--dump`	Displays detailed information regarding configuration files for a package installed on the system (when used following the `-q` and `-c` options)
`-e` `--erase`	Removes a specified package from the system
`-F` `--freshen`	Upgrades a specified package only if an older version exists on the system
`-f` `--file`	Displays the package to which the specified file belongs (when used with the `-q` option)
`-h` `--hash`	Prints hash marks on the screen to indicate installation progress (when used with the `-i` option)
`-i` `--install`	Installs a specified package (provided the `-q` option is not used)
`-i` `--info`	Displays full information about the specified package (when used with the `-q` option)

Table 11-8: Common options used with the rpm utility

Working with the Red Hat Package Manager (RPM)



Option	Description
`-K`	When used before a filename argument, validates the checksum listed within the RPM file
`-l` `--list`	Lists the filenames the specified package comprises (when used with the `-q` option)
`--nodeps`	Forces the RPM to avoid checking for dependencies before installing packages (when used following the `-i` option)
`-q` `--query`	Queries information about packages on the system
`--test`	Performs a test installation only (when used with the `-i` option)
`-U` `--upgrade`	Upgrades a specified package; the package is installed even if no older version exists on the system
`-V` `--verify`	Verifies the location of all files that belong to the specified package
`-v`	Prints verbose information when installing or manipulating packages

Table 11-8 (cont'd): Common options used with the rpm utility

Working with RPM

- Commands:
  - Install package
    - `rpm -ivh <package.rpm>`
      - `i: install, v: verbose, h: print progress`
  - Check dependencies
    - `rpm -qpR <package.rpm>`
    - `q: query, p: package capabilities, R: dependencies`
  - Check existing package
    - `rpm -q <package.rpm>`
  - Check installed packages
    - `rpm -qa (query all)`
      - Add --last to show recently installed

(fluxbox example)

Working with Yum

- Install package

  -
    ```
    yum install <package>
    ```
- Remove package

  -
    ```
    yum remove <package>
    ```
- Update package

  -
    ```
    yum update <package>
    ```
- Search

  -
    ```
    yum search <package>
    ```
- List all available

  -
    ```
    yum list | less
    ```

Yum cheatsheet: https://access.redhat.com/articles/yum-cheat-sheet

Groupinstall

- Handy to pull in all usual resources:
- For a dev environment:
  - sudo yum groupinstall “Development tools”
- To see all:
  - yum grouplist

Gnome Package Manager

- The GNOME (and KDE) desktop environments contain a utility that periodically checks for updated RPM packages
- GUI utility:
  - Check for updates manually
  - Search for and install new RPM software packages
  - Manage installed RPM software packages
- System → Administration → Add/Remove SW

Working with the Debian Package Manager (DPM)

- The Debian Package Manager is used by default on Linux distributions based on the Debian Linux distribution
  - Such as Ubuntu
- DPM packages
  - Use the .deb extension
  - Installed and managed using the `dpkg` command
- DPM database
  - Stored within files in the /var/lib/dpkg directory)

Working with the Debian Package Manager (DPM)

- To download and install DPM packages
  - Use the `apt-get` command

- To remove a DPM package
  - Use the `apt-get remove` or `apt-get purge` command

- To update and upgrade a DPM package
  - Use the `apt-get update` and `apt-get` upgrade command

Working with the Debian Package Manager (DPM)

- You can add new repositories using the `add-apt-repository` command
  - Or search available repository information using the `apt-cache` command

Summary

- Linux software:
  - Compile from source
  - Add pre-compiled binaries
  - Use a package manager

- Package Managers
  - Install and manage compiled software
  - RPM
    - Red Hat Package Manager
  - DPM
    - Debian Package Manager

In-Class Work

1. What are **two** “best practices” for logging?

2. Consider your term project:
   - Based on what you have worked on thus far, what are **two services** that should be logged?
   - What kind of information will you get out of logging each service?
   - What is one way that you can **automatically** get notified of a critical error?
   - What are some **packages** that you can install (using YUM or RPM) to assist you in developing your project?

Class Over

- HW5 due next Wednesday (10/14) at 11:59pm.

Creating an RPM file

- Spec file
  - Contains all the information to build and install a package
- Requires
  - What does this package need in order to run?
    - e.g., some other package? certain (version of a) library?
- Provides: What does this package provide?
  - e.g., binary, library, etc.

Create RPM

- Need development tools (of course)

```
sudo yum install @development-tools #Compiler
sudo yum install fedora-packager #RPM packaging
sudo yum install elfutils-devel  #Create applications
```

Create RPM

- Create new user to handle RPM creation
  - Don't want to do it as root – build process may cause far-reaching issues

```
sudo adduser rpmuser
sudo passwd rpmuser
```

- Login as user
- Setup directory structure

```
rpmdev-setuptree
ls
```

Create RPM

- Make sample executable

hello_world.c

```
#include <stdio.h>

int main(int argc, char* argv[])
{
  printf(“Hello CIT348.  Isn’t this fun?!!??!”);
  return 0;
}
```

Create RPM

- Make spec file

hello_world.spec (template auto-created!)

Day 2

- https://www.youtube.com/watch?v=-fQGPZTECYs