

Welcome to Linux

Lecture 1.1

What is Linux?

Leo Ufimtsev

Leonidas@RedHat.com

Reference: [LINK](#)



Some history

- 1969 - the Unix operating system by Ken Thompson and Dennis Ritchie
- Unix became widely adopted by academics and businesses
- 1977 - the Berkeley Software Distribution (BSD) by UC Berkeley. A lawsuit *USL v. BSDi*.
- 1983 – the GNU project by Richard Stallman - a free UNIX-like operating system (GPL). GNU incomplete – no kernel
- 1991- Linus Torvalds, an undergraduate student from Finland, began a “just for fun” project that later became the Linux kernel.

Linus Torvalds : Famous first message 1991

Hello everybody out there using minix -

I'm doing a (free) operating system (just a hobby, **won't be big and professional like gnu**) for 386(486) AT clones. This has been brewing since april, and is starting to get ready. I'd like any feedback on things people like/dislike in minix, as my OS resembles it somewhat (same physical layout of the file-system (due to practical reasons) among other things).

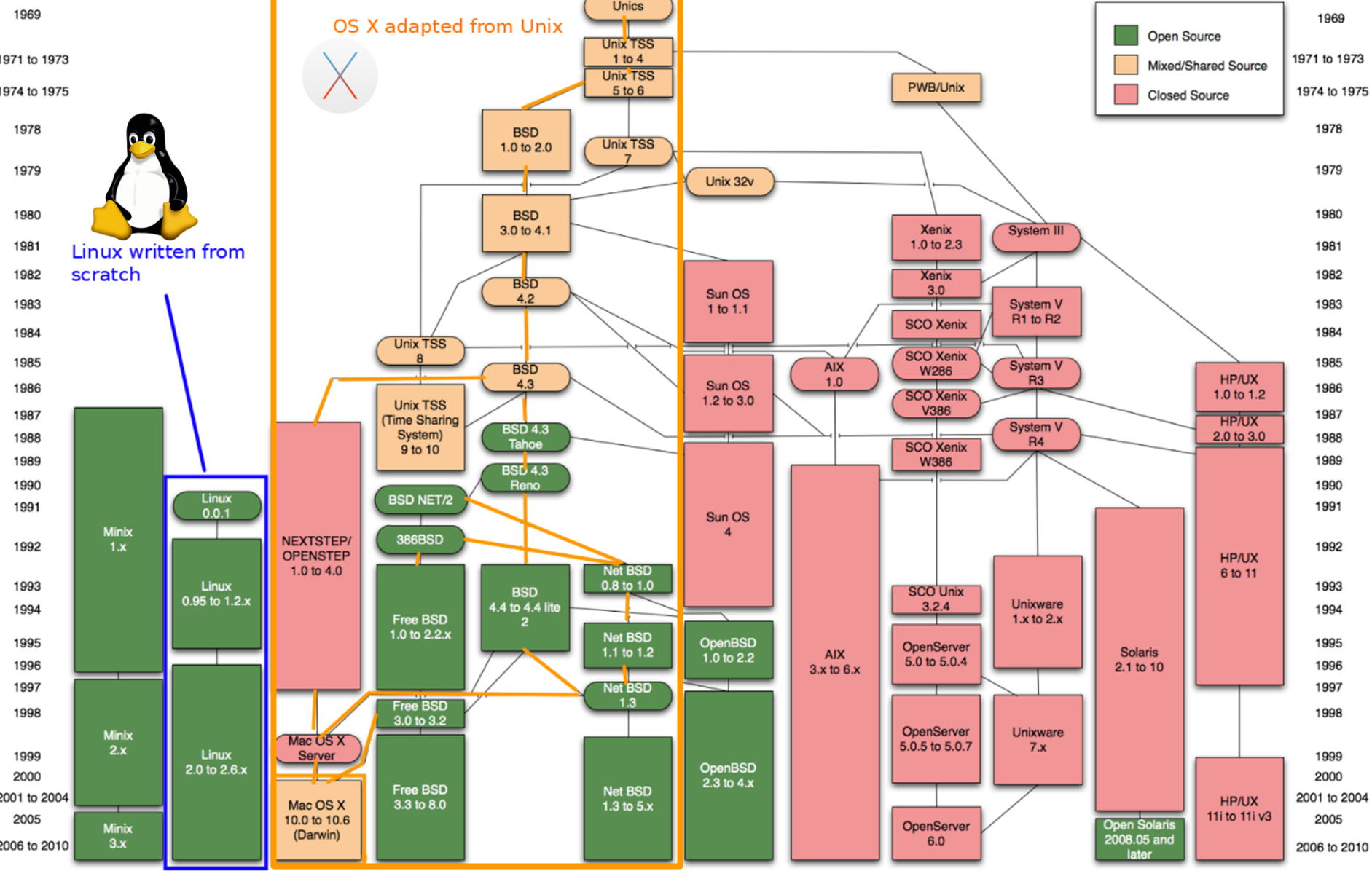


Linus (torvalds@kruuna.helsinki.fi)

PS. Yes - it's free of any minix code, and it has a multi-threaded fs. It is NOT portable (uses 386 task switching etc), and it probably never will support anything other than AT-harddisks, as that's all I have :-).

This guy also made git btw

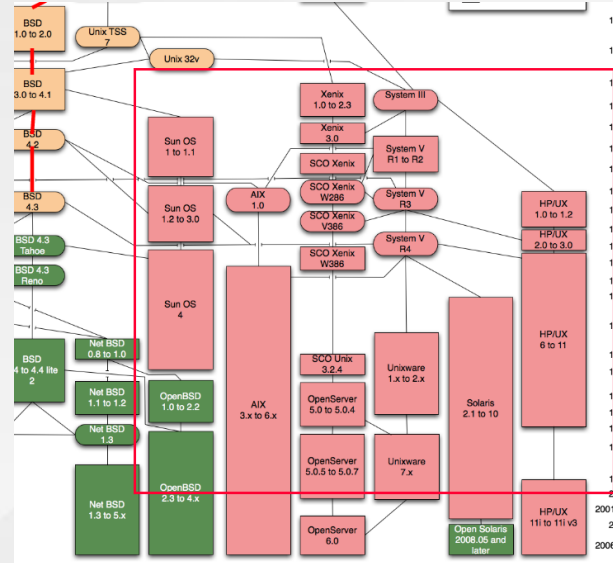
— Linus Torvalds[14]



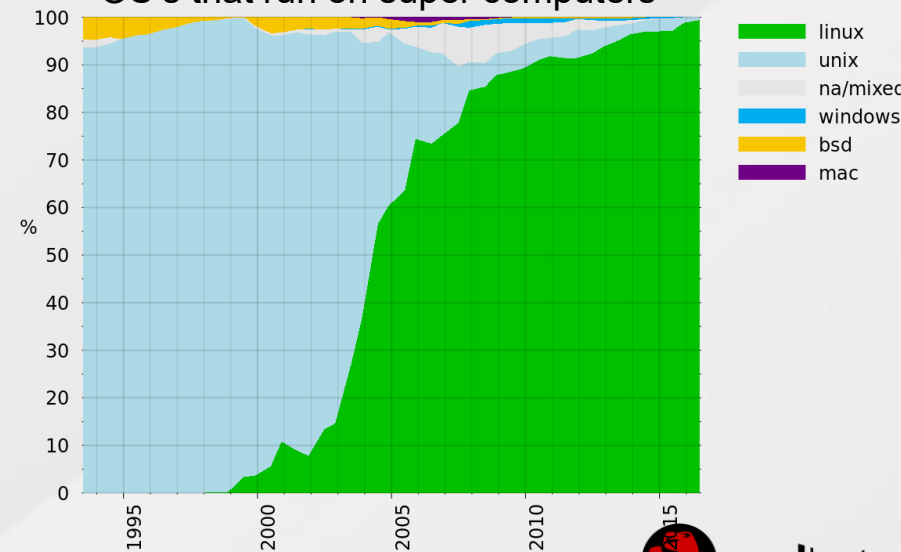
Linux written from scratch

Linux vs Unix

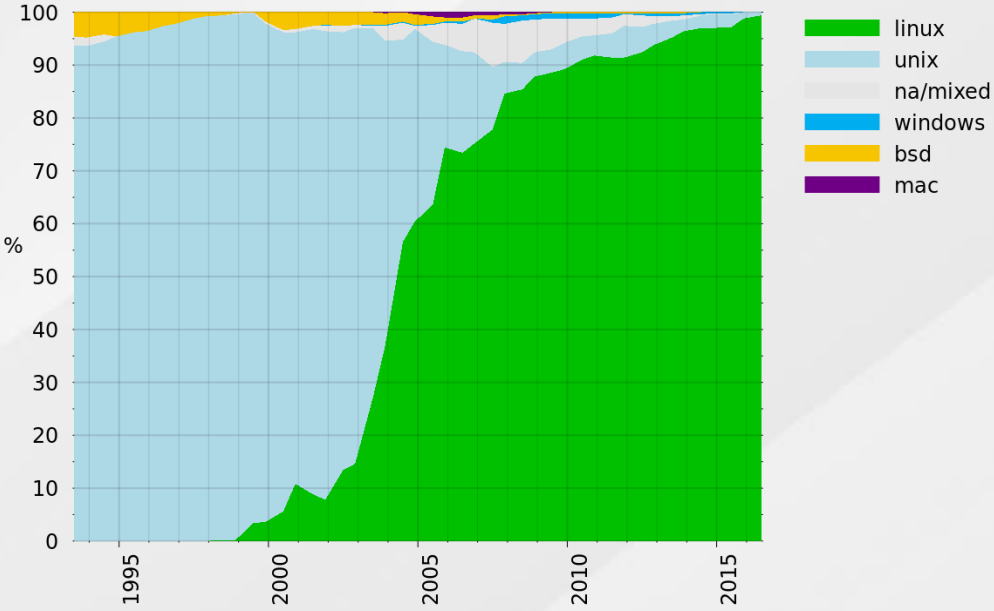
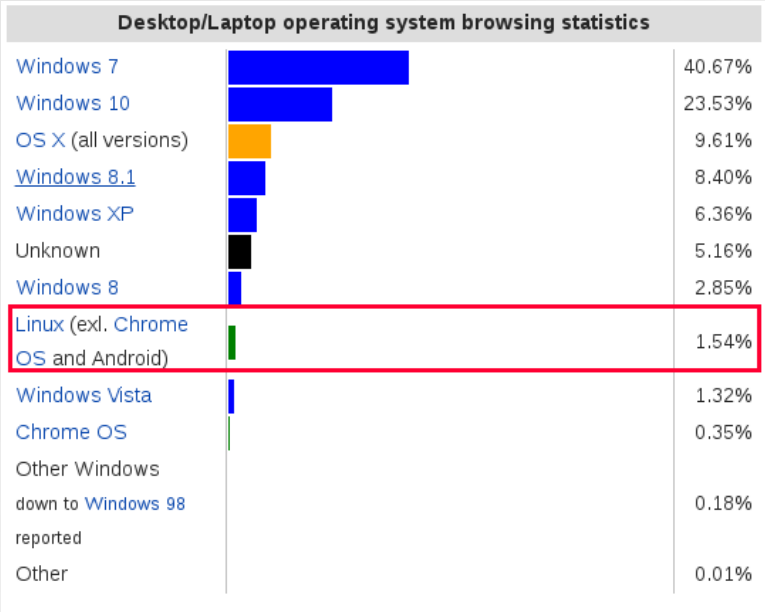
- Unix was often **closed source**, companies sued each other over features.
- Linux was open source, designed for anyone to modify and add enhancements.
- Unix was for big commercial servers.
- Linux was initially used for personal computers, but then turned out to be useful for all kinds of computers, especially servers.



OS's that run on super computers



Desktop vs Server



Linus 2012: “I started Linux as a desktop operating system. And it's the only area where Linux hasn't completely taken over. That just annoys the hell out of me.”

Vision of Open Source

Transparency, Inclusivity, Adaptability

Greater agility (your feature can run on other machines tomorrow)

Faster innovation (Ideas gather momentum quickly by community)

Increased engagement (Everyone is responsive for the whole project, not just your niche bit of code)

Open Source Software

- Source code made available with a license in which the copyright holder provides the rights to **study, change, and distribute** the software to anyone and for any purpose

Examples of open source

- VLC media player
- Eclipse
- Git
- 83%+ of web servers run Apache or Nginix
- Popular programming languages:
 - Python
 - OpenJDK Java
 - C/C++
- **Not open source:**
 - Oracle Java -> Free to download and use for commercial purposes, but not allowed to re-distribute.

// Oracle tends to sue friends. Treat with care.

“Oracle will seek a staggering \$9.3 billion in 2nd trial against Google” <http://arstechnica.com/tech-policy/2016/03/oracle-will-seek-a-staggering-9-3-billion-in-2nd-trial-against-google/>



Microsoft goes open source..

- In 2006, Microsoft referred to Open Source as “Cancer”. But things have changed
- Microsoft attended Linux Conference 2016 in Toronto
- Open sourced .Net (2015)
(ex: Visual Studio written in .Net)
- Open sourced PowerShell (2016)
- Azure runs Linux
 - 20% of Azure’s servers run Linux.
- Microsoft isn’t becoming a fully ‘open source’ company, (It’s unlikely Windows will be open source), but they’re building bridges...



How does open source make money?

- Normally you buy the software and support is free (maybe not so much anymore)
- In Open Source you get the software for free, but you can (optionally) pay for support and services. Or a service is free for opensource/personal use, but requires a fee for commercial use (e.g GitHub).
- Sort of like you get a car for free, but you pay for maintenance of the car and for someone to teach you how to drive it.

note: Maintenance starts on day 1.



The “Git” technology is free and open source. You can run your own git on your own server. Hosted solutions exist to serve your code for you, ex GitHub, Bitbucket, GitLab.

Windows / OSX

- Focused on ease of usability, “Off the shelf software ready for use”. Very nice desktop U.I.
- Designed for specific hardware. Runs on limited system configurations.
- Designed for a specific target audience. Limited OS customization.



Linux

- Focused on development “Tool box and materials, for you to build your thing”. Very nice set of development tools.
- Designed to be flexible, to run on just about anything.
- Designed so it can be customizable to suit one’s needs.
- Not that user friendly.



Interchange existing modules

- Like Lego, build what you need -> Highly customizable and flexible.
- Linux provides great components that you can glue together and great tools to put things together with.
- There is satisfaction in running something you build yourself.



Micro systems,
embedded
systems
With limited
resources



Real time systems
that don't tolerate
delay in processing
(Linux can be
customized to do
so)



Super-scale
computers with
1000's of
nodes.

Now Linux runs on almost anything

- Mars rover
- Playstation 3
- NASA control systems
- US department of defense.
 - (Humvees have 4 onboard Red Hat servers *1)
- Navy Submarines
- Ikea/Walmart
- New York stock exchange
- Raspberry pi
- Android
- Google's entire server fleet runs Linux. Amazon / Wikipedia etc..
- Cern / ~97% of all super computers.

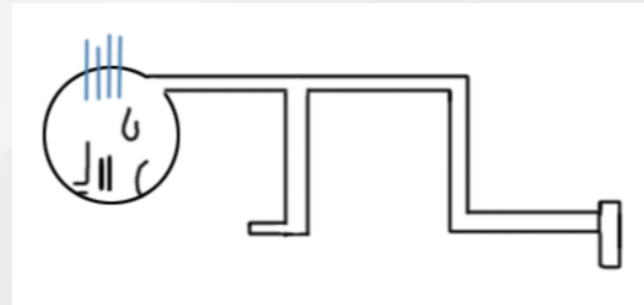


*1: <https://www.redhat.com/en/about/press-releases/142>

When there is a problem

Commercial:

- Complain, Hope, OTL



OTL -> Desperation / disappointment

Open Source:

- Complain, Hope, OTL
**OR fix/implement
yourself.**

Tux!

- Linus was at a Zoo, where he saw penguins
- Later he found the picture (on the right) on an FTP site
- Logo was made using GIMP
- (T)orvalds (U)ni(X)"

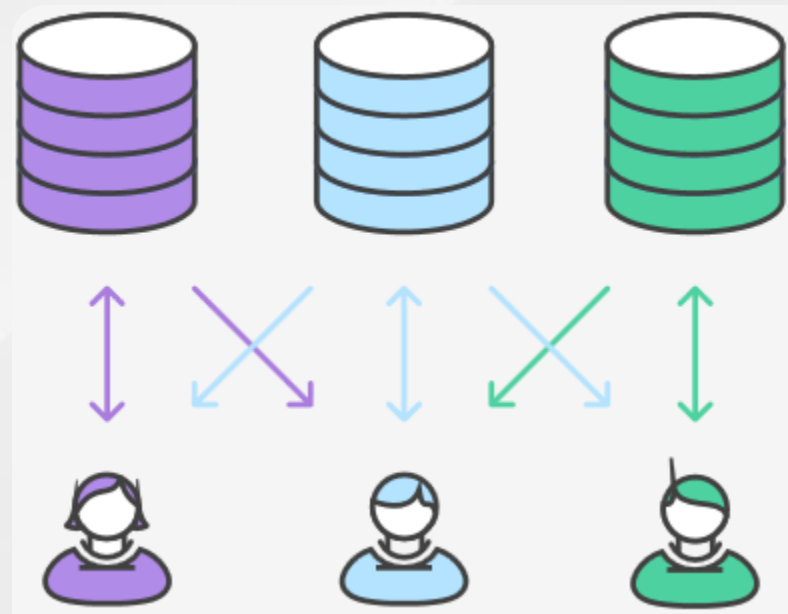


Original image that
Inspired Linus



Forking things






- If community doesn't agree with **your idea**, but you really like your idea, fork it!
- If nobody maintains a **project** anymore and you want to **revive** it, fork it!
- If you want to build a new product based on existing project, fork it!



Can lead to a lot of variants

542 linux distributions known to public

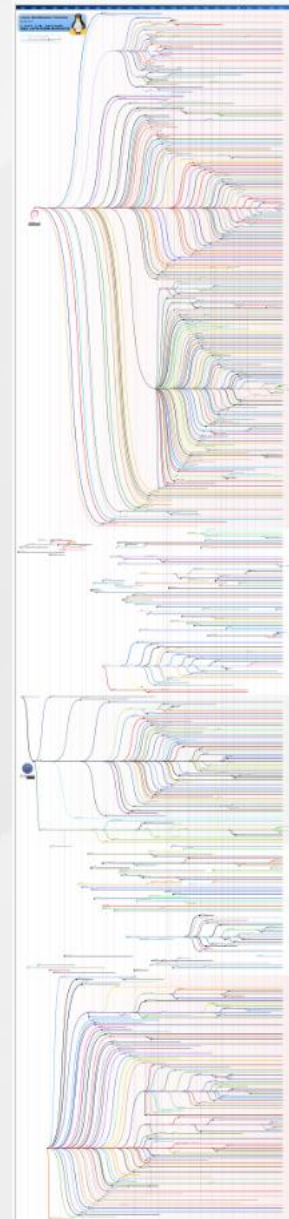
Most forked from major distros:

- Debian 
 - Ubuntu 
 - ChromeOS
 - SteamOS
- Android
 - Cyanogen OS (ex OnePlus)
- Slackware
 - OpenSUSE 
- Arch
 - Black arch 
- Red Hat 
 - Fedora (Upstream Red Hat, for dev work)
 - CentOS (Free Red Hat clone)
 - Oracle Linux



We use this on our
developer machines

https://en.wikipedia.org/wiki/List_of_Linux_distributions



Good ideas transcend their original hardware/OS/programming language

- Ex: Make

- Made in 1977 by intern Stuart Feldman at Bell Labs
- Completely re-written over the years, in different languages
- Still widely used today



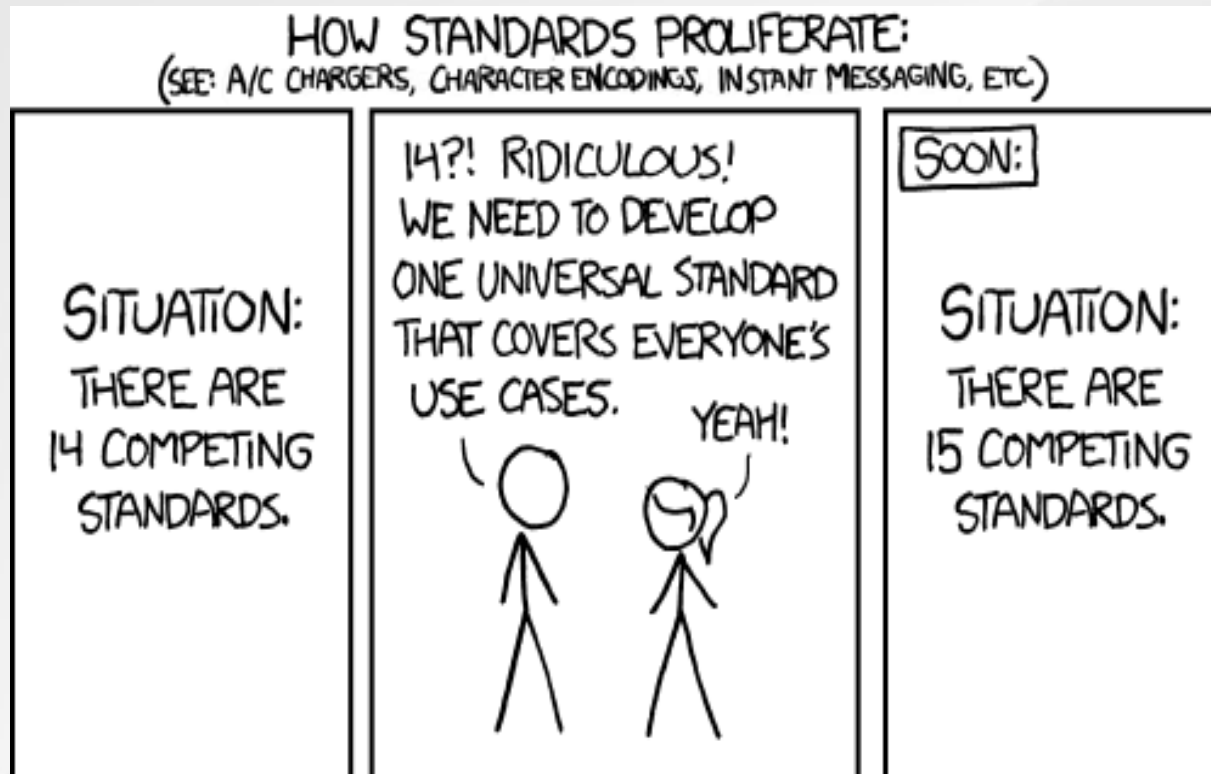
```
[~]$ make
```

- Ex: Emacs

- Made in 1976.
- Transcended original hardware, OS and was rewritten in a different programming language. (Has it's own programming language actually)
- Today so big that it has it's own StackExchange site (Sublime, Atom etc don't..).



A lot of diversity & standards ...



Open communication

- Freedom of speech is very important
 - In Red Hat there is a “Memo List”, where anyone can email the entire company & share opinion.
- People express their opinions
- Your reputation (rather than seniority) decides how people treat you.
- Some people are polite, some are less polite;
> feelings can get hurt.



Linus cares a lot about Linux. So if someone submits a really bad patch and doesn't admit the patch is bad, things can get rough.

From Linus Torvalds <> **Date** Sun, 23 Dec 2012 09:36:15 -0800

Subject Re: [Regression w/ patch] Media commit causes user space to misbahave (was: Re: Linux 3.8-rc1)

On Sun, Dec 23, 2012 at 6:08 AM, Mauro Carvalho Chehab
<mcchehab@redhat.com> wrote:

>
> Are you saying that pulseaudio is entering on some weird loop if the
> returned value is not -EINVAL? That seems a bug at pulseaudio.

Mauro, SHUT THE FUCK UP!

It's a bug alright - in the kernel. How long have you been a maintainer? And you **still** haven't learnt the first rule of kernel maintenance?

If a change results in user programs breaking, it's a bug in the kernel. We never EVER blame the user programs. How hard can this be to understand?

To make matters worse, commit f0ed2ce840b3 is clearly total and utter CRAP even if it didn't break applications. ENOENT is not a valid error return from an ioctl. Never has been, never will be. ENOENT means "No such file and directory", and is for path operations. ioctl's are done on files that have already been opened, there's no way in hell that ENOENT would ever be valid.

> So, on a first glance, this doesn't sound like a regression,
> but, instead, it looks tha pulseaudio/tumbleweed has some serious
> bugs and/or regressions.

Shut up, Mauro. And I don't *_ever_* want to hear that kind of obvious garbage and idiocy from a kernel maintainer again. Seriously.

I'd wait for Rafael's patch to go through you, but I have another error report in my mailbox of all KDE media applications being broken by v3.8-rc1, and I bet it's the same kernel bug. And you've shown yourself to not be competent in this issue, so I'll apply it directly and immediately myself.

WE DO NOT BREAK USERSPACE!

Seriously. How hard is this rule to understand? We particularly don't break user space with TOTAL CRAP. I'm angry, because your whole email was so *_horribly_* wrong, and the patch that broke things was so obviously crap. The whole patch is incredibly broken shit. It adds an insane error code (ENOENT), and then because it's so insane, it adds a few places to fix it up ('ret == -ENOENT ? -EINVAL : ret').

The fact that you then try to make **excuses** for breaking user space, and blaming some external program that **used** to work, is just shameful. It's not how we work.

Fix your f*cking "compliance tool", because it is obviously broken. And fix your approach to kernel programming.

Linus

But normally he's quite a nice guy 😊.

A few of his quotes:

“I'm always right. This time I'm just even more right than usual.” - 2005

“I have an ego the size of a small planet, but I'm not `_always_ right [...]`.” 2007

“The memory management on the PowerPC can be used to frighten small children.” - 2012

“We don't merge kernel code just because user space was written by a retarded monkey on crack.” - 2015

“I like offending people, because I think people who get offended should be offended.” - 2012



Contributing to Open Source

- Use open source programs.
 - Find something that bothers you, try to fix it.
- Find a project you're interested in
 - Look at starred projects in GitHub
 - Look at things you use
- Work on it
 - Learn the process for that project
 - Submit patches
- Good experience to put on your CV





Example: Contribute bugfix to Mozilla

- Takes about 1-2 (maybe 3) weekends to set things up and fix a bug.
- Mozilla has a “Good first bug list”
https://wiki.mozilla.org/Good_first_bug
- Bugs available for most languages (C++/Javascript/java/Python/HTML & CSS)
- Lots of “Contributing to” articles:
https://developer.mozilla.org/en-US/docs/Mozilla/Developer_guide/Introduction
- Folks are very helpful, find them on IRC/ mailing lists
- General process:
 - Find an easy-ish bug, check out source code.
 - Ask where to look for bug, people will help you where to navigate in code base
 - Fix bug, submit patch. Await feedback. Improve fix from feedback until it’s polished.

Let's learn some Linux

Scope for the first 2 weeks

- Get familiar with Linux
- Use existing utilities with CLI
- Write simple utilities in C

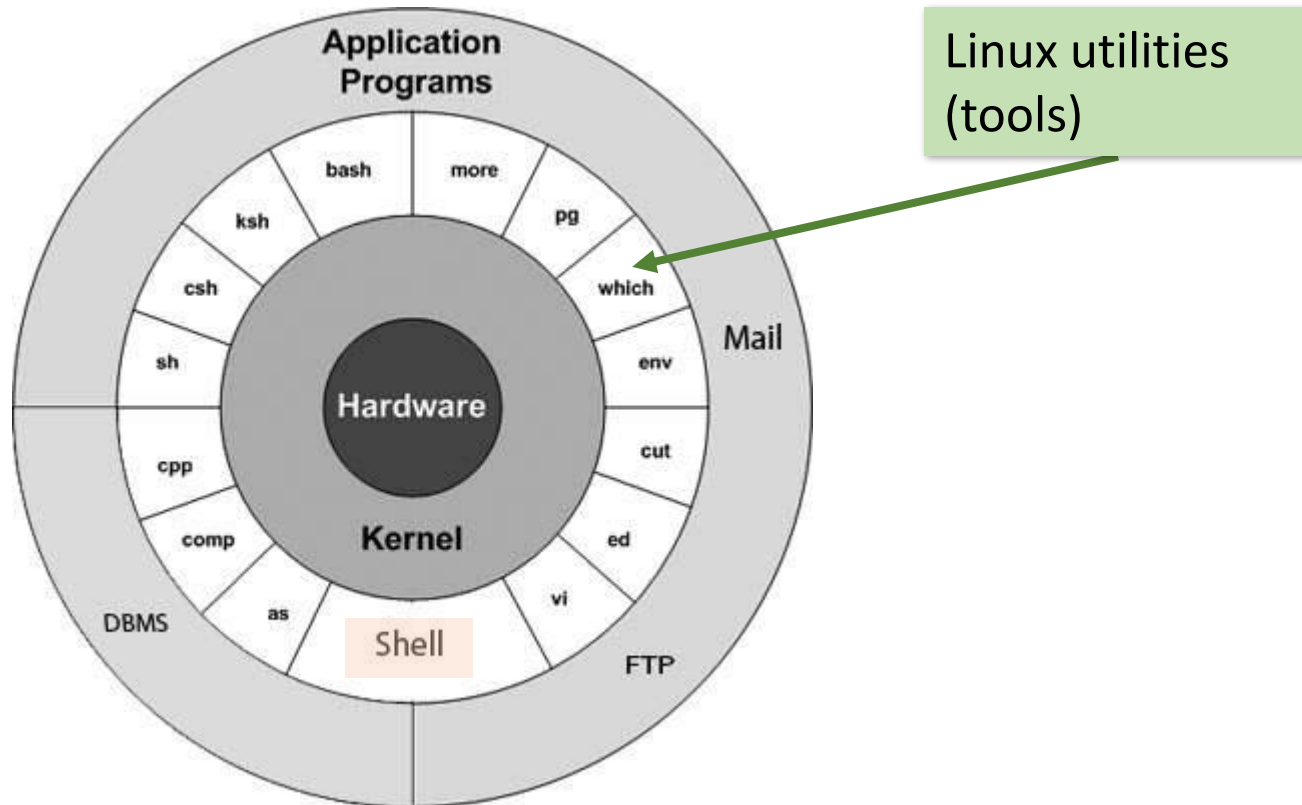
Scope for the rest of the course

- Write application programs which **interact with Linux kernel**
- All this using **C programming language**

Linux operating system

- Open-source
- Written in portable yet highly efficient language
- Built-in networking
- Built-in multitasking
- Rich software development environment
- Open interface to kernel
- Powerful and flexible CLI (command-line interface)

Linux structure



Linux kernel

- **Process** creation, and scheduling multiple processes
- **Memory** management: allocation, release
- **File system** on disk: abstraction over physical disk blocks
- Access to I/O devices: **device drivers**
- **Networking**: routing and exchange of messages
- Interface for user programs to perform requests to kernel:
system calls

Linux file system

```
ls -li
```

File abstraction

- “Everything is a file.”
- **Unified file interface** = **open**, **read**, **write**, **close** for
 - regular files
 - directories
 - devices
 - video
 - keyboard
 - network

Index node - inode

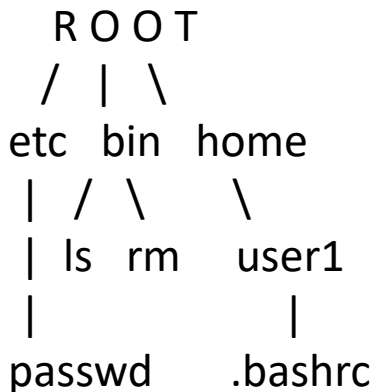
- The actual data for each Unix file is managed by numbered on-disk data structures called *inodes*
- One inode is allocated for each file and each directory
- Unix inodes have unique numbers, not names, and it is these numbers that are kept in directories alongside the names.

```
ls -i
```

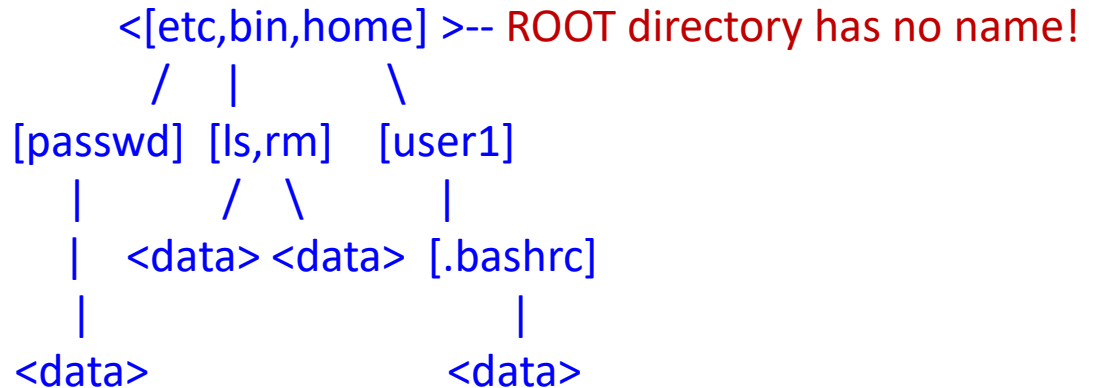
Typical Linux file hierarchy

- Everything starts in the “root” directory
- A **directory** is a file that contains directory entries: pairs of (child name, inode).

=====



=====



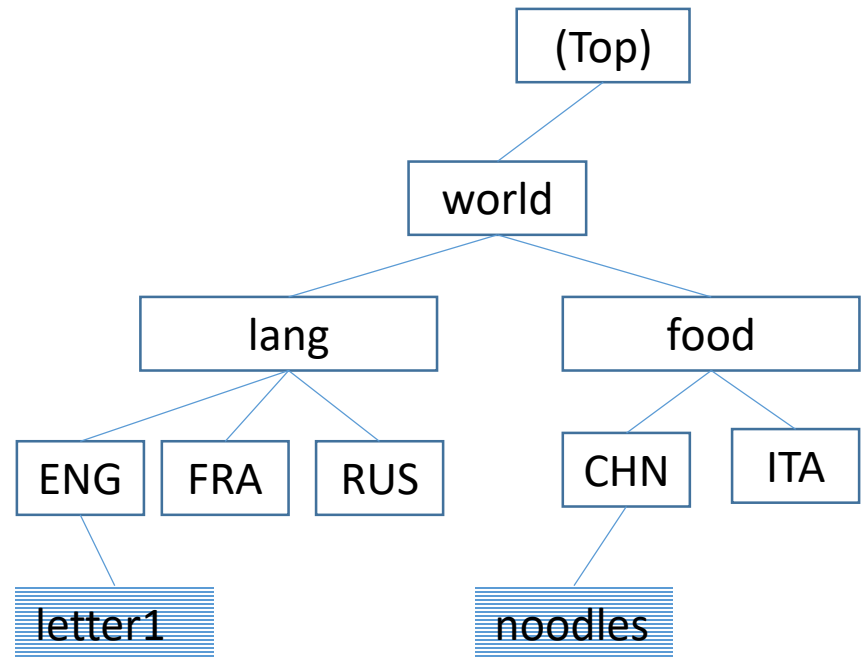
What is stored in inode

- A list of pointers to the disk blocks that belong to that file or directory.
- The attributes of the file or directory (permissions, size, access/modify times, etc.); but, not the name of the file or directory:

inodes have only **numbers**, **attributes**, and **disk blocks** – **not names**. The names are kept separately, in parent directories

What is stored in inodes - example

i	directory	What is stored
8	top	[world-9, ...]
9	world	[lang-10, food-11]
10	lang	[ENG-12, FRA-16, RUS-17]
11	food	[CHN-13, ITA-18]
12	ENG	[letter1-14]
13	CHN	[noodles-15]
14	letter1	File data blocks
15	noodles	File data blocks

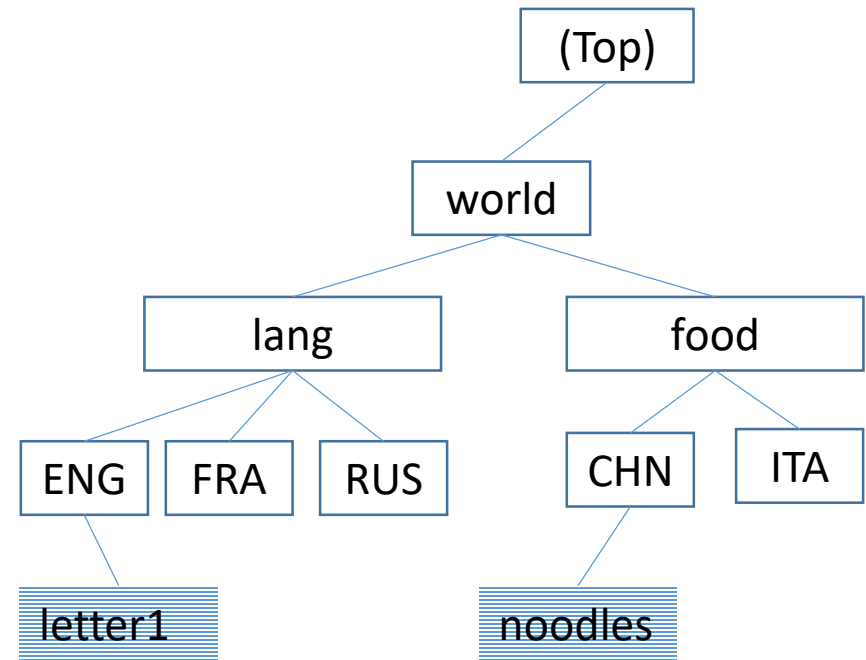


File vs. directory inodes

- *File inode* stores location of disk blocks containing file data
- *Directory inode* stores location of disk blocks with lists of names and inode numbers.
- You must use the inode number from the directory to find the inode on disk to read its attribute information; reading the directory only tells you the name and inode number.
- Hence the difference between **ls** and **ls -l**

What is NOT stored in inodes?

i	directory	What is stored
8	top	[world-9, ...]
9	world	[lang-10, food-11]
10	lang	[ENG-12, FRA-16, RUS-17]
11	food	[CHN-13, ITA-18]
12	ENG	[letter1-14]
13	CHN	[noodles-15]
14	letter1	File data blocks
15	noodles	File data blocks



The name of a file is NOT stored in file inode – it is stored in the parent directory

File data can be recovered (without name)

- The name and inode number pair in a directory is the only connection between a name and the thing it names on disk
- If a directory is damaged, the names of the things are lost and inodes become “orphan”
- The things themselves may be undamaged. You can run a file system recovery program such as **fsck** to recover the data (but not the names)

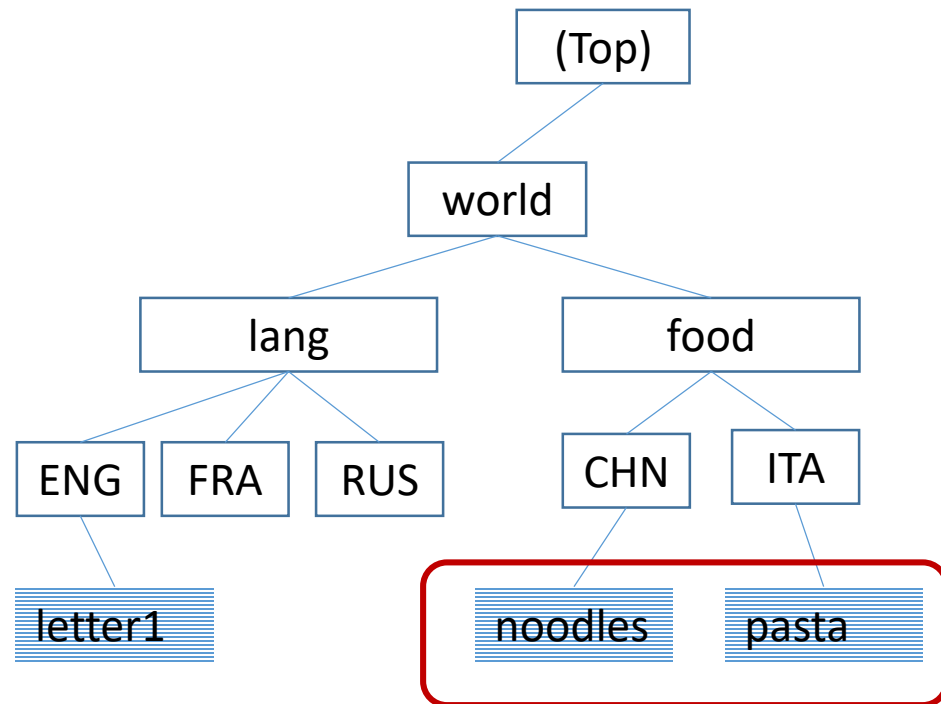
Multiple names to the same file: hard links

- An entry in a directory file which specifies a pair of (name, inode) is called a **hard link**.
- There can be several hard links to the same physical file!

```
ln bar foo  
ls -li
```

Hard link example

i	directory	What is stored
11	food	[CHN-13, ITA-18]
12	ENG	[letter1-14]
13	CHN	[noodles-15]
14	letter1	File data blocks
15	noodles	File data blocks
16		...
17		...
18	ITA	[pasta-15]



2 names of the same file



```
cd world/food
```

```
ln CHN/noodles ITA/pasta
```

Tracing inodes example: /home/alex/foobar

```
+-----+
#2 |. 2 |.. 2 | home 5 | usr 9 | tmp 11 | etc 23 | ... |
+-----+
|   | The inode #2 above is the ROOT directory. It has the
|   | name "home" in it. The *directory* "home" is not
|   | here; only the *name* is here. The ROOT directory
|   | itself does not have a name!
|   |
V
+-----+
#5 |. 5 |.. 2 | alex 31 | leslie 36 | pat 39 | abcd0001 21 | ... |
+-----+
|   | The inode #5 above is the "home" directory. The name
|   | "home" isn't here; it's up in the ROOT directory,
|   | above. This directory has the name "alex" in it.
|   |
V
+-----+
#31|. 31|.. 5 | foobar 12 | temp 15 | literature 7 | demo 6 | ... |
+-----+
|   | The inode #31 above is
|   | the "alex" directory. The
|   | name "alex" isn't here;
|   | it's up in the "home"
|   | directory, above. This
|   | directory has the names
|   | "foobar" and "literature"
|   | in it.
|   |
V
+-----+
#7 |. 7 |.. 31| | barfoo 12 | morestuff 123 | junk 99 | ... |
+-----+
|   | The inode #7 above is the "literature" directory.
|   | The name "literature" isn't here; it's up
|   | in the "alex" directory. This directory has
|   | the name "barfoo" in it.
|   |
V
V
*-----* This inode #12 on the left is a file inode.
| file data | It contains the data blocks for the file.
#12 | file data | This file happens to have two names, "foobar"
| file data | and "barfoo", but those names are not here.
*-----* The names of this file are up in the two
          directories that point to this file, above.
```

What else is stored in inodes?

- Directory inode stores two additional (name,inode) pairs:
 - Itself:  – number
 - Parent:  – number
- Both file and directory inodes store *permissions*

Directories cannot have hard links!

- Files may have many names ("links") - but directories cannot!
- Each directory inode is allowed to appear **once** in exactly one parent directory and no more.
- This restriction means that every sub-directory only has one parent directory, and that means the special name ".." (dot dot) in a sub-directory always refers unambiguously to its unique parent directory.
- This directory linking restriction prevents loops and cycles in the file system tree, preventing cases where a sub-sub-directory might contain a link back up to a parent directory.

ln vs. ln -s

- Storage Space: no new inodes with hard links - in soft links we create a new inode to store the path to the file
- Performance: directly accessing the disk pointer instead of going through the path stored in soft link file.
- Renaming target file: the hard link will still work, but soft link will point to the previous file location.
- Redundancy: with hard link, the data is safe, until all the links to the file are deleted - in soft link, you will lose the data if the master instance of the file is deleted.

Programmable shell

Running built-in utilities

Shells

- Special-purpose programs designed to read **commands** typed by the user and **shell scripts**, interpret them, and execute appropriate programs in response
- Many shells, i.e.:
 - Bourne shell (SH)
 - Bourne again shell (**BASH**)

How the shell is collaborating with the kernel

- Shell:
 - accepts command names and arguments as input
 - finds the executable
 - interprets the arguments
 - loads an executable into memory and hands it off to the OS to run.
- Kernel:
 - starts the process of executing the program

How does shell know where to find an executable

- PATH variable: List of directories to be consulted when looking up commands specified without path names.
- E.g. you type "cat", it execs "/bin/cat". It finds it by looking through the path, which is a list of directories including /bin.

```
sh: PATH=/bin:/usr/bin
```

```
sh: PATH=/bin:/usr/bin: 
```

Recording your session

- script

Working with files

- ls
- pwd
- cp
- mv
- Compress: bzip2, tar

Globbering

- **Globbering** - process of expanding a non-specific file name containing a wildcard character into a set of specific file names that exist
- **Standard wildcards (globbering patterns)**
 - * matches any number of any character
 - ? matches any one character
 - [range] :
 - `m[a, o, u]m`, `m[a-d]m`
 - {} matches at least one (or):
 - `cp {*.doc, *.pdf} ~`
 - [!] excluding
 - `rm myfile[!9]`

Sharing files:
permissions

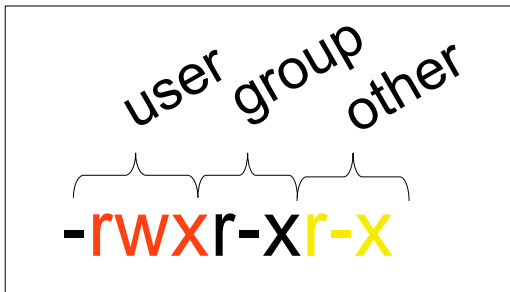
Users belong to user groups (up to 16-32 groups max)

wolf:~% **groups** mgbarsky

mgbarsky : instrs csc209h csc343h csc443h cs209hi
cs343hi cs443hi

Permissions as numbers

Number	Octal Representation	Ref
0	No permission	---
1	Execute permission	--X
2	Write permission	-W-
3	Execute and write permission: 1 (execute) + 2 (write) = 3	-WX
4	Read permission	r--
5	Read and execute permission: 4 (read) + 1 (execute) = 5	r-X
6	Read and write permission: 4 (read) + 2 (write) = 6	rw-
7	All permissions: 4 (read) + 2 (write) + 1 (execute) = 7	rwX



Setting permissions

- `chmod 755 <filename>`
 - 3 numbers between 0 and 7, the octal value for that category of user
 - Quiz — what is the command to set the permissions of the file *classlist* to be world readable but writeable only by the file owner and members of the group.
- Or using:
 - `chmod u+rwx`
 - `chmod go-x`
 - `chmod a=x`
 - adds or removes permissions for those categories of users

Special permissions: root

- There is a special user called root, which have all the permissions for all the files in the system
- sudo



File Permissions

chmod (change mode)

- Changes the permissions (mode) on an existing inode (file, directory, etc.)

ls -lid (list structure, long version, inode, directory)

- Shows the permissions of an inode

Working with file contents

- cat
- less
- head
- tail
- sort
- uniq
- wc
- comm
- diff
- grep
- cut

grep

- Searching plain-text data sets for lines matching a regular expression.
- Main uses:
 - *grep -x* matches entire line
 - *grep -v* matches all lines which do not contain a pattern
 - *grep ^pattern* – matches lines which start with ‘pattern’

grep examples

```
wolf:~/w2% grep craig faculty
```

```
wolf:~/w2% grep Craig faculty
```

Craig, Michelle

Berns, Craig

Craigin, Pete

```
wolf:~/w2% grep ^Craig faculty
```

Craig, Michelle

Craigin, Pete

Reg. expressions for *grep*

- ^ (Caret) = match expression at the start of a line, as in ^A.
- \$ (Question) = match expression at the end of a line, as in A\$.
- \ (Back Slash) = turn off the special meaning of the next character, as in \^.
- [] (Brackets) = match any **one** of the enclosed characters, as in [aeiou]. Use Hyphen "-" for a range, as in [0-9].
- [^] = match any one character **except** those enclosed in [], as in [^0-9].
- . (Period) = match a single character of any value, except end of line.
- * (Asterisk) = match zero or more of the preceding character or expression.
- \{x,y\} = match **x** to **y** occurrences of the preceding.
- \{x\} = match exactly **x** occurrences of the preceding.
- \{x,\} = match **x** or more occurrences of the preceding.

Output redirection

- If the notation `> file` is appended to any command that normally writes its output to standard output, the output of that command will be written to file:

```
who > users
```

Input redirection

- The commands that normally take their input from standard input can have their input redirected from a file:

```
wc -l users
```

```
wc -l < users
```

Processes

Kernel starts a process for each program

To see all the processes:

```
ps
```

PID	TTY	TIME	CMD
26357	pts/5	00:00:00	tcsh
26558	pts/5	00:00:00	bash
32624	pts/5	00:00:00	ps

Process groups and pipelining

- Connect processes, by letting the standard output of one process feed into the standard input of another. That mechanism is called a *pipe*.
- Connecting simple processes in a pipeline allows to perform complex tasks without complex programs.

```
$ls -l | sort -k5n | less
```

Displays files in current directory sorted by file size

ls -l command – output fields: total

```
wolf:~/w2% ls -l
```

```
total 8
```

```
-rw-r--r-- 1 mgbarsky instrs 94 Apr 11 10:20 faculty
```

```
-rw-r--r-- 1 mgbarsky instrs 76 Apr 11 10:20 hello.c
```


```
wolf:~/w2% ls -s
```

```
total 8
```

```
4 faculty 4 hello.c
```

```
wolf:~/w2%
```

Total number of disk
blocks used by all files in
this directory



ls -l command – output fields: col 1

```
wolf:~/w2% ls -l
```

```
total 8
```

```
-rw-r--r-- 1 mgbarsky instrs 94 Apr 11 10:20 faculty
```

```
-rw-r--r-- 1 mgbarsky instrs 76 Apr 11 10:20 hello.c
```



File permissions

ls -l command – output fields: col 1

```
wolf:~/w2% mkdir a
```

```
wolf:~/w2% ls -l
```

```
total 12
```

```
drwx----- 2 mgbarsky instrs 4096 May 15 14:28 a
```

```
-rw-r--r-- 1 mgbarsky instrs  94 Apr 11 10:20 faculty
```

```
-rw-r--r-- 1 mgbarsky instrs  76 Apr 11 10:20 hello.c
```

File type

-: regular file

d : directory

s : socket

p : pipe

D : Door

l : symbolic link etc.

Lynux considers all *regular files* with **x** permissions executable

x permission for *directory* means you can search subfolders in the current directory

ls -l command – output fields: col 2

```
wolf:~/w2% mkdir a
```

```
wolf:~/w2% ls -l
```

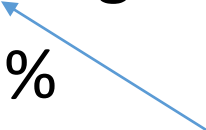
```
total 12
```

```
drwx----- 2 mgbarsky instrs 4096 May 15 14:28 a
```

```
-rw-r--r-- 1 mgbarsky instrs  94 Apr 11 10:20 faculty
```

```
-rw-r--r-- 1 mgbarsky instrs  76 Apr 11 10:20 hello.c
```

```
wolf:~/w2%
```



Number of hard
links to the
current file in the
entire system –
link count

You can delete regular file
only if link count becomes
0 after deleting

ls -l command – output fields: col 3

```
wolf:~/w2% mkdir a
```

```
wolf:~/w2% ls -l
```

```
total 12
```

```
drwx----- 2 mgbarsky instrs 4096 May 15 14:28 a
```

```
-rw-r--r-- 1 mgbarsky instrs  94 Apr 11 10:20 faculty
```

```
-rw-r--r-- 1 mgbarsky instrs  76 Apr 11 10:20 hello.c
```

```
wolf:~/w2%
```



User

ls -l command – output fields: col 4

```
wolf:~/w2% mkdir a
```

```
wolf:~/w2% ls -l
```

```
total 12
```

```
drwx----- 2 mgbarsky instrs 4096 May 15 14:28 a
```

```
-rw-r--r-- 1 mgbarsky instrs  94 Apr 11 10:20 faculty
```

```
-rw-r--r-- 1 mgbarsky instrs  76 Apr 11 10:20 hello.c
```

```
wolf:~/w2%
```

User group



ls -l command – output fields: col 5

```
wolf:~/w2% mkdir a
```

```
wolf:~/w2% ls -l
```


```
total 12
```

```
drwx----- 2 mgbarsky instrs 4096 May 15 14:28 a
```

```
-rw-r--r-- 1 mgbarsky instrs 94 Apr 11 10:20 faculty
```

```
-rw-r--r-- 1 mgbarsky instrs 76 Apr 11 10:20 hello.c
```

```
wolf:~/w2%
```



Size of directory or file in bytes – in some systems allocates at least 1 block for the directory

ls -l command – output fields: col 6

```
wolf:~/w2% mkdir a
```

```
wolf:~/w2% ls -l
```

```
total 12
```

```
drwx----- 2 mgbarsky instrs 4096 May 15 14:28 a
```

```
-rw-r--r-- 1 mgbarsky instrs 94 Apr 11 10:20 faculty
```

```
-rw-r--r-- 1 mgbarsky instrs 76 Apr 11 10:20 hello.c
```

```
wolf:~/w2%
```

Last modified



ls -l command – output fields: col 7

```
wolf:~/w2% mkdir a
```

```
wolf:~/w2% ls -l
```

```
total 12
```

```
drwx----- 2 mgbarsky instrs 4096 May 15 14:28 a
```

```
-rw-r--r-- 1 mgbarsky instrs  94 Apr 11 10:20 faculty
```

```
-rw-r--r-- 1 mgbarsky instrs  76 Apr 11 10:20 hello.c
```

```
wolf:~/w2%
```



File name

At home

Recording your session

- `script <session1>`
- To stop recording: `exit`

To get help

- `man <program_name>`
- `<program_name> --help`
- `info coreutils '<program_name> invocation'`

Working with files

- ls
- pwd
- cp
- mv
- rm
- Compress: bzip2, tar
- wget <url>

Working with file contents

- cat
- less
- head
- tail
- sort
- uniq
- wc
- comm
- diff
- grep
- cut
- file (dos2unix, unix2dos)

- vimtutor
 - ESC
 - :q