CS2211b

Software Tools and Systems Programming

Week 1b

Introduction to UNIX

Announcements

TAs

Evan Debenham
edebenha@uwo.ca

Kun Xie
kxie5@uwo.ca

Consulting Hours:

Thursday 4:30PM to 5:30PM
In MC244
Start on January 25th

Tuesday 4:30PM to 5:30PM
In MC244
Start on January 16th

GAUL Guest Account

Username: cs2211

Password: cs2211-2018

Expires 1 week after add/drop date

- In case you have not yet received your GAUL password

Introduction to UNIX/Linux

Western Science

What Is An Operating System?

Without an OS

What do you do with just computer hardware?

- If someone gives you a computer with no software whatsoever, how do you get it to do anything?
- You write a program that runs on the hardware

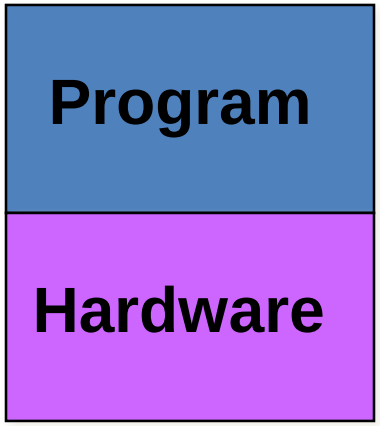
In the early days, that was the way it worked ...

- You started with just the bare hardware
- You wrote a program that did everything:
  - Including managing all aspects of the hardware
  - Including solving your particular problem

Your program was all the computer did!

What Is An Operating System?

Without an OS

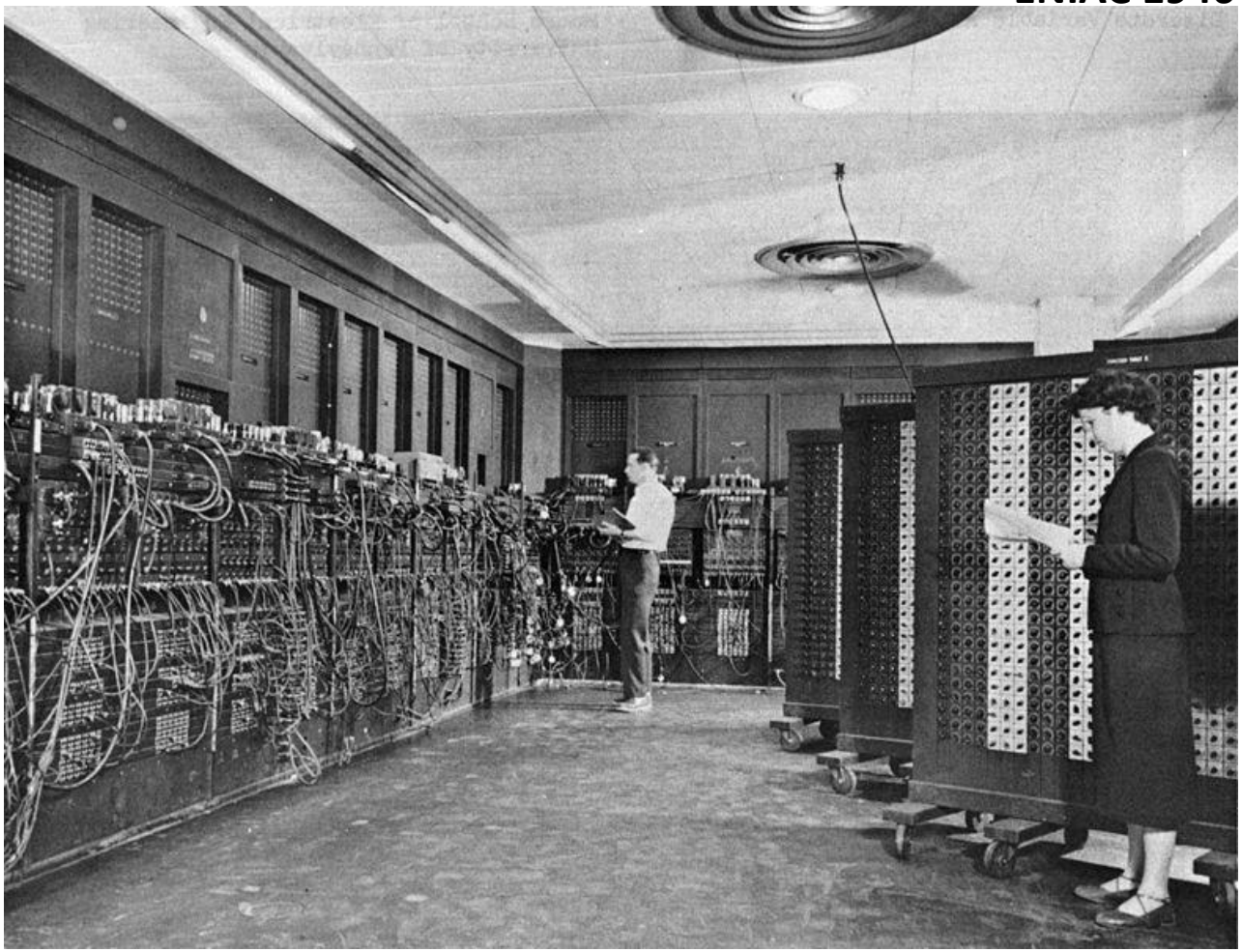


- Each program runs directly on the hardware
- Each program must do everything
- Each program needs to know the details of the hardware and how to use it
- If the hardware changes, the program must change as well
- The hardware supports only one program at a time - each user must wait until the previous program is done to “share” the hardware with other users.
- Writing programs is incredibly complex and expensive

What Is An Operating System?

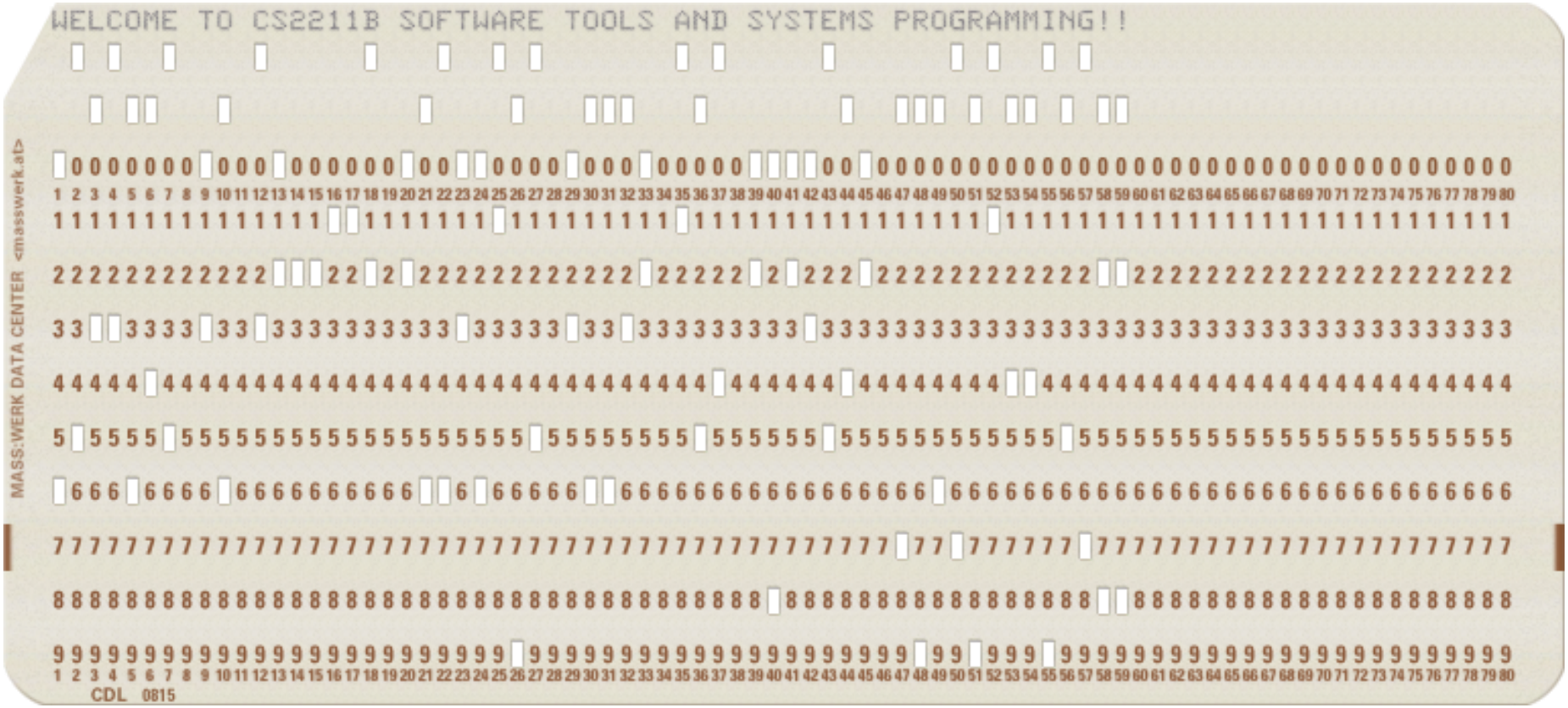
40s to mid 50s

ENIAC 1946



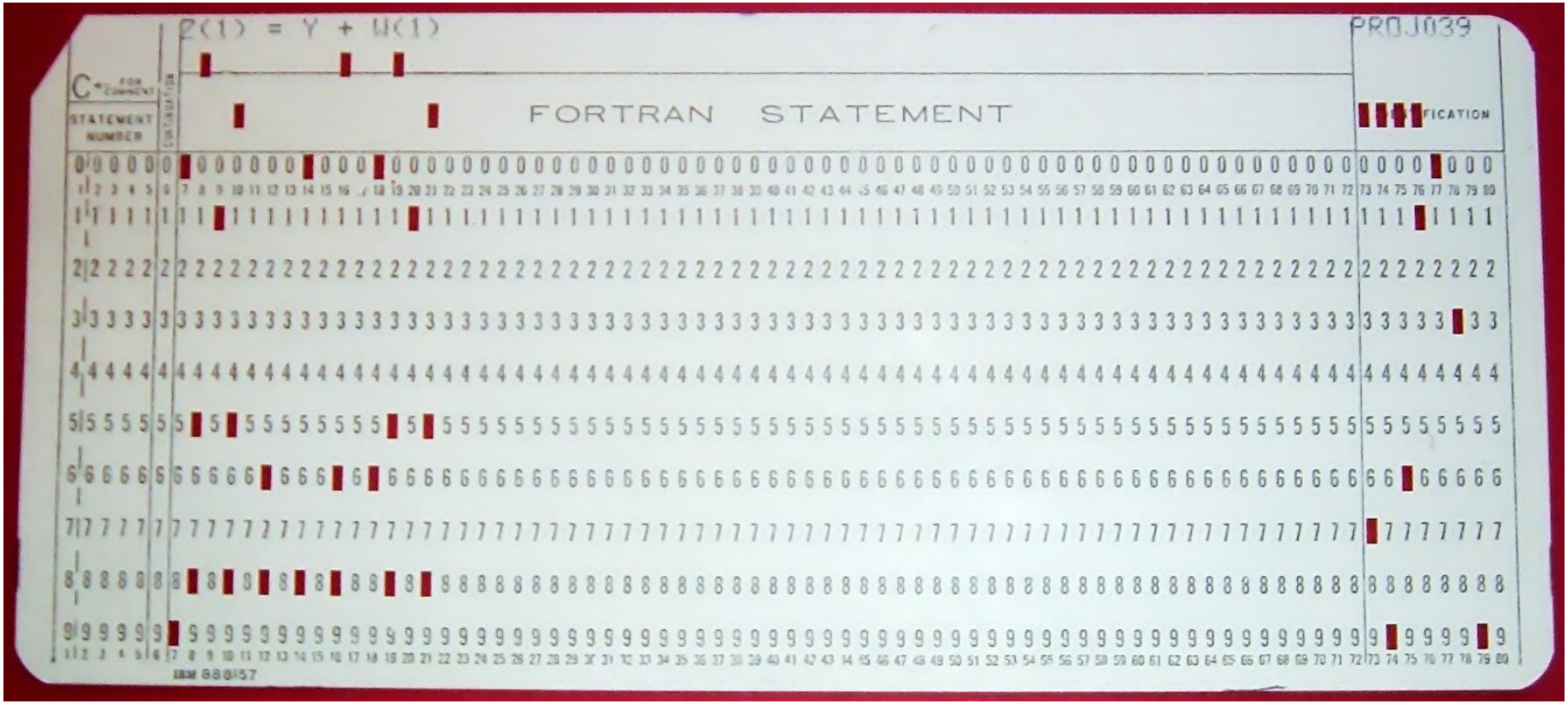
What Is An Operating System?

Mid 50s to mid 60s



What Is An Operating System?

Mid 50s to mid 60s



What Is An Operating System?

IBM Type 29 Card Punch (1964)

What Is An Operating System?

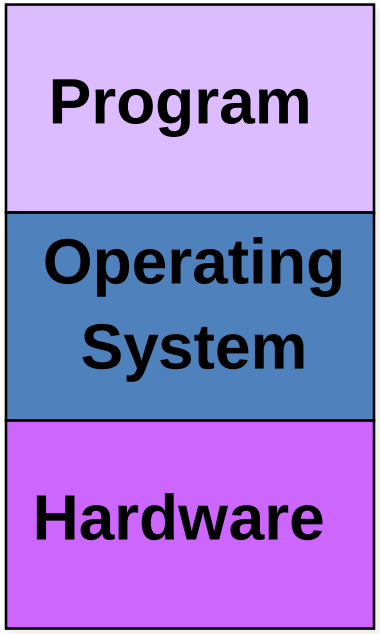
Demo

The Virtual Keypunch

Site: http://www.masswerk.at/keypunch

What Is An Operating System?

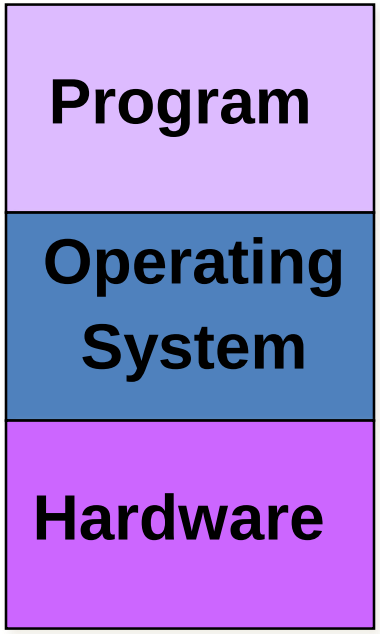
With an OS



- Operating system runs directly on the hardware
- Operating system is in charge of managing the hardware
- Operating system hides the details of hardware from software - provides a much simpler interface for programs

What Is An Operating System?

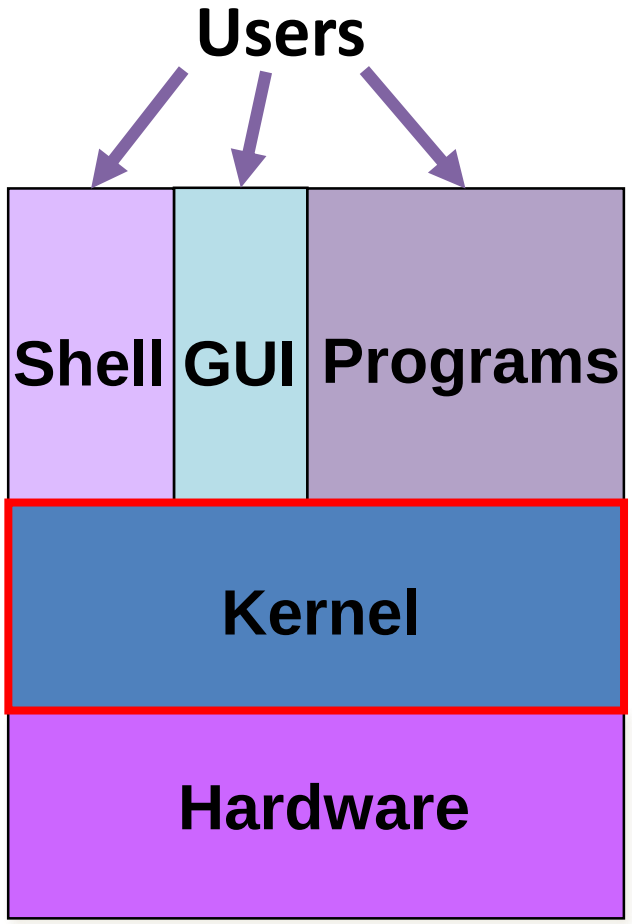
Benefits of OSs



- An operating system hides the hardware
  - Programs are portable
  - Programs are hardware-independent
- An operating system manages the computer
  - Programming is easier
  - Using a computer is easier - you no longer need to program or be an expert to use it
- An operating system facilitates resource sharing
  - Several programs can run at once
  - Simultaneous users

What Is An Operating System?

Parts of an OSs



- Divided into two parts
  - Kernel
  - Shell
- The kernel is the core of the operating system
  - Interacts with the hardware
  - Programs communicate with the kernel to access the hardware
  - Manages memory, schedules processes, decides priorities and other tasks

What Is An Operating System?

Kernel Types

Monolithic Kernels

- One very large program
- Performs everything by it's self
- Fast and efficient
- Difficult to design and maintain

UNIX, UNIX-like, Linux, BSD, DOS, etc.

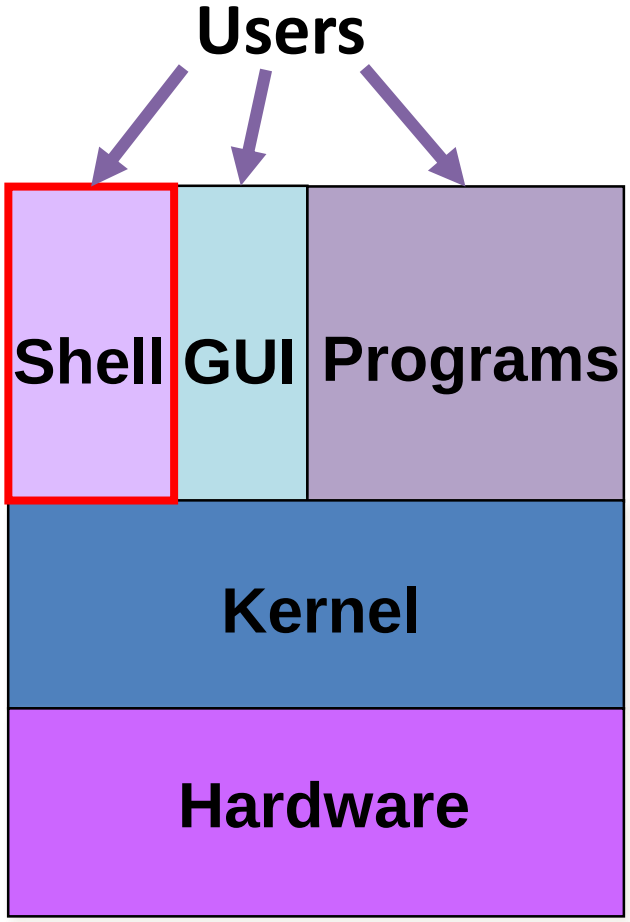
Microkernels

- A much smaller program
- Performs the most basic tasks only
- To perform more functions, calls upon a set of other programs
- Slower and less efficient
- Easier to design and maintain

Some embedded systems, mostly no longer used.

What Is An Operating System?

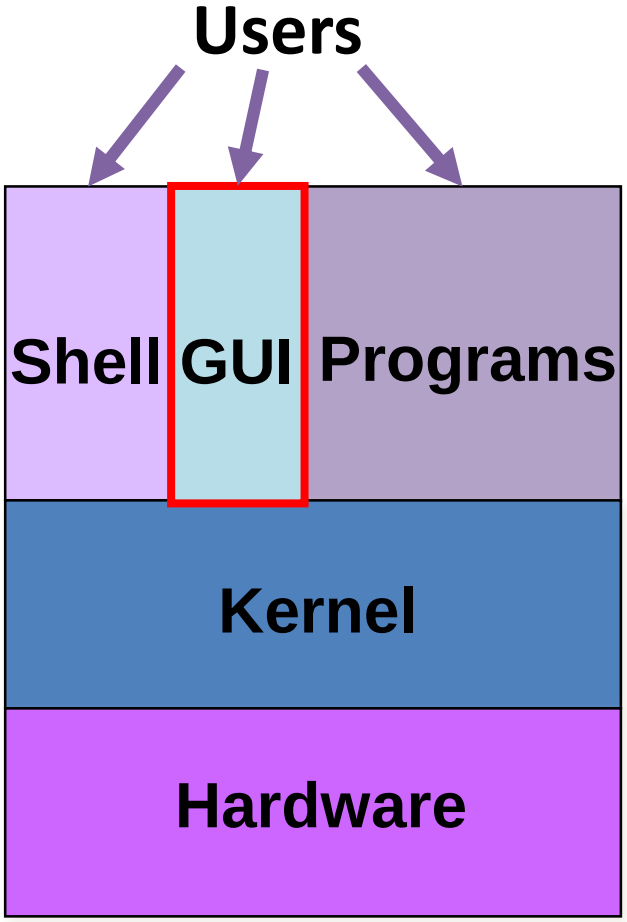
Parts of an OSs



- The shell is a special type of program (a command processor) that acts as an interface to the system for users
  - Interface between users and the kernel
  - Can have several shells but only one kernel running at a time
  - Shell interprets text based commands from the user
  - Examples UNIX/Linux shells:
    - sh
    - csh
    - bash
    - tcsh

What Is An Operating System?

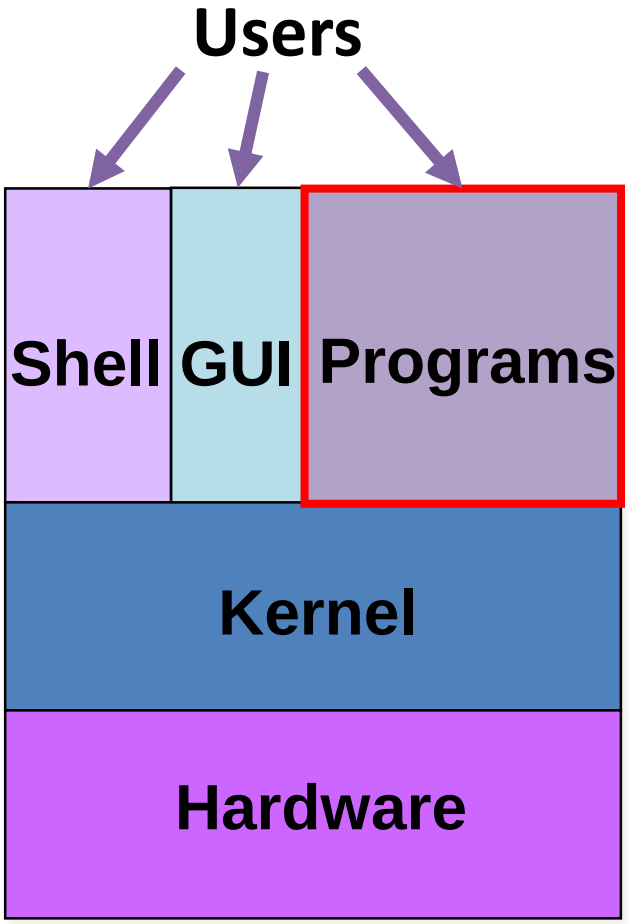
Parts of an OSs



- Optionally an OS can have a Graphical User Interface (GUI)
  - Provides graphical interface for users to interact with programs and the kernel
  - Essentially a mouse/graphics based shell
  - Makes OS usable by non experts
  - For UNIX/Linux, a popular GUI is the X Window System

What Is An Operating System?

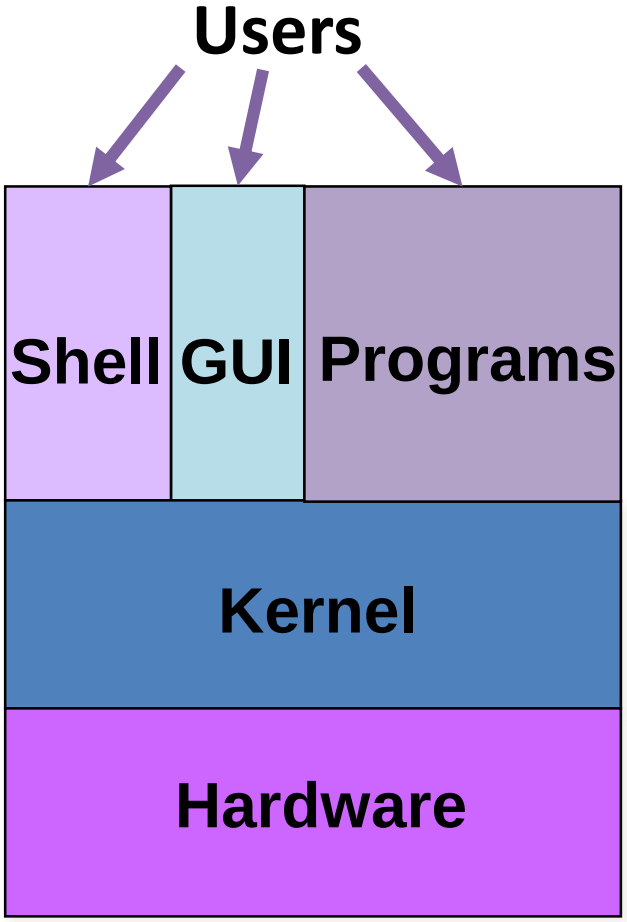
Parts of an OSs



- OS may be prepackaged with many utility programs
  - Utility programs allow users to do things like edit files, communicate with each other, develop programs, etc.
  - UNIX examples: who, wc, grep, man
- Users may also install or create their own programs (user programs)
- Programs can use other programs, shell commands or call the kernel directly.

What Is An Operating System?

A UNIX OS



- A UNIX OS generally has…
  - A single kernel running at a time
  - One or more shells
  - GUI is optional
  - Prepackaged utility programs
  - A monolithic kernel
  - Multiple users simultaneously (multiuser)
  - Multiple programs running simultaneously (multitasking)

In-class Activity

In groups of 4 or 5 answer the following question:

Based on what you have read in the assigned readings, seen in this lecture and your experience with UNIX/Linux so far, **what differences have you noticed between Windows and UNIX/Linux?**

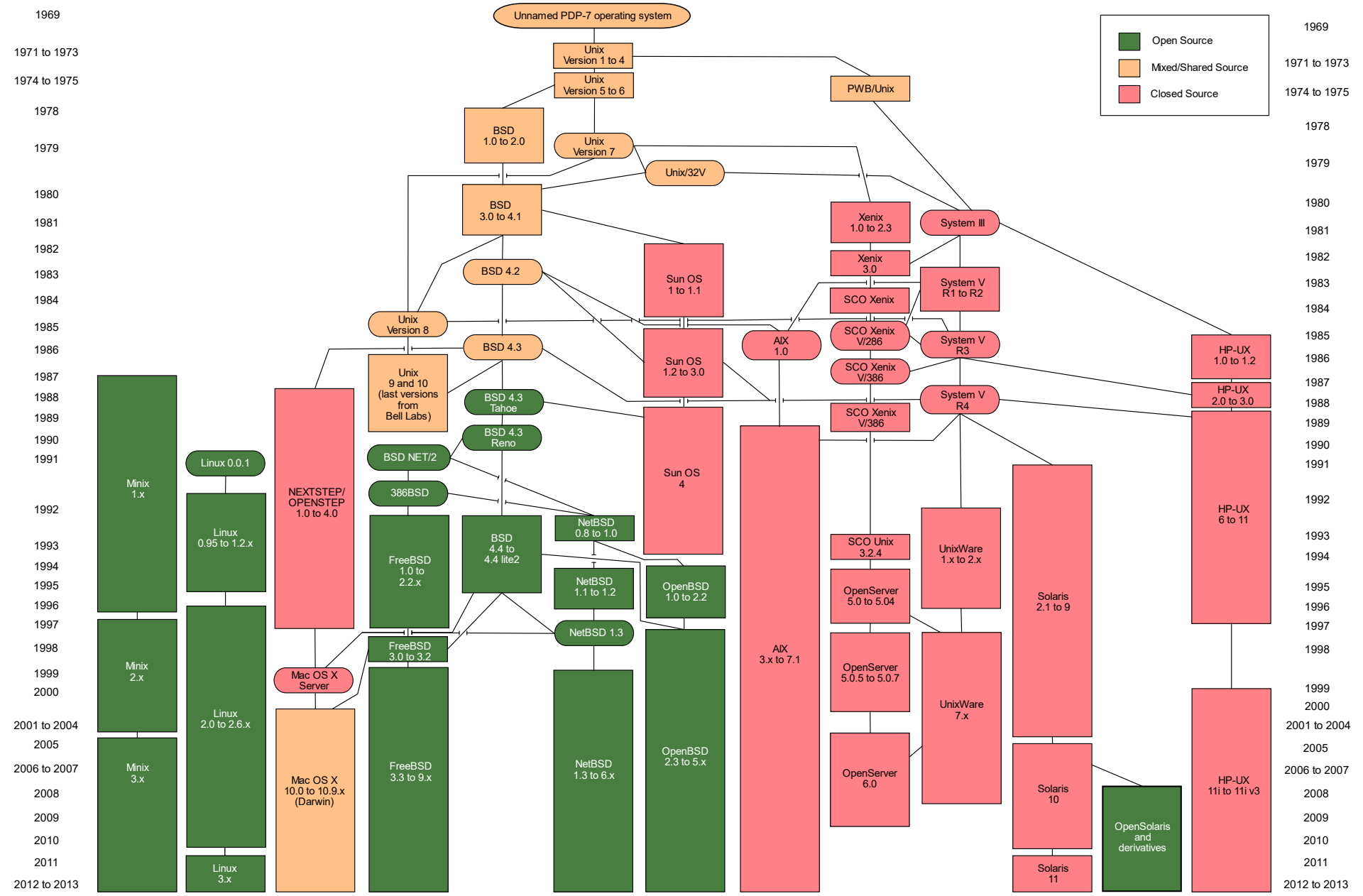
Windows vs. UNIX/Linux

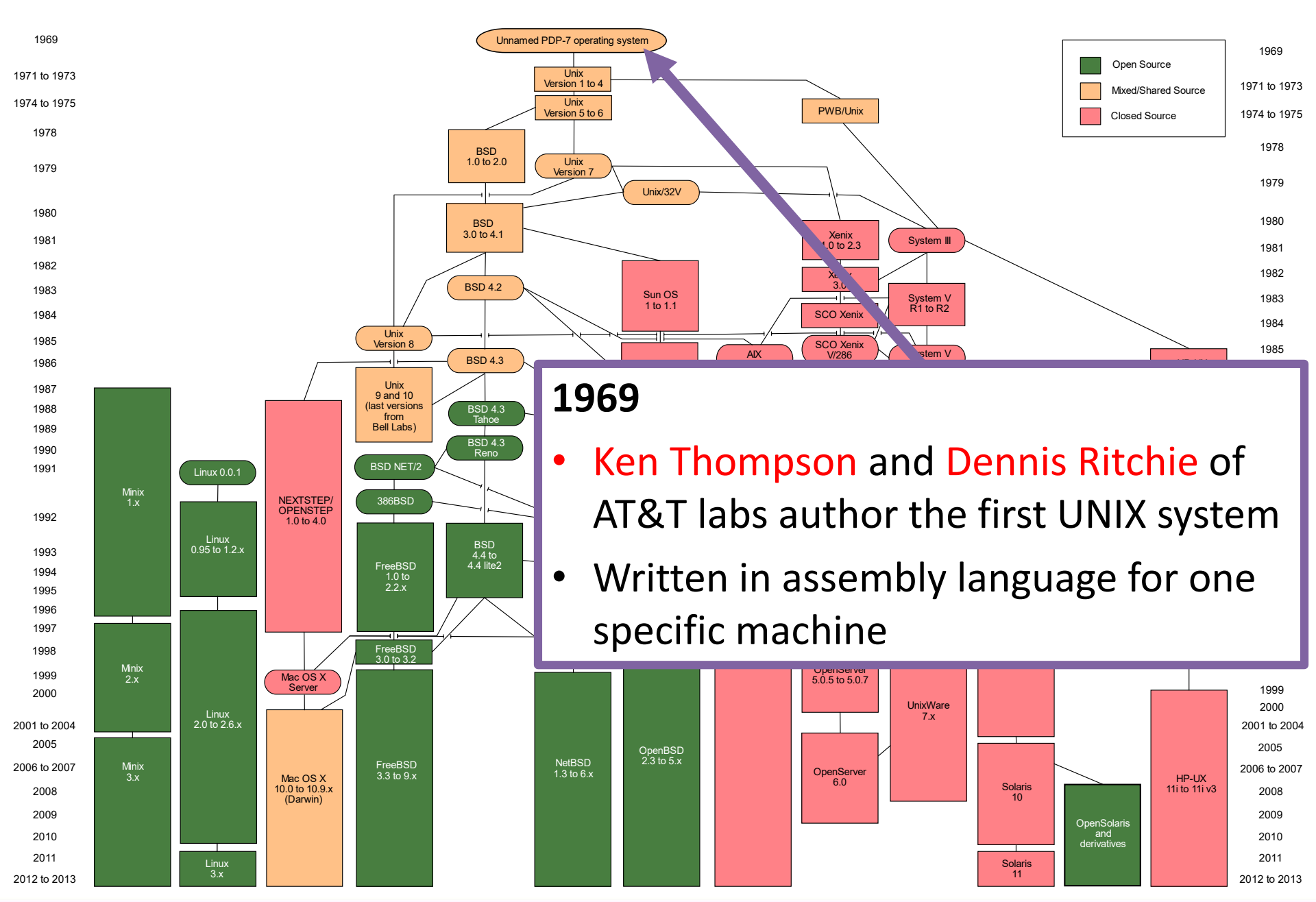
	Windows	UNIX/Linux
License	Closed source	Some open source & free
GUI	Built-in	Optional
Interface	GUI	Shell
Users	Single-user (at a time)	Multiuser
Included Programs	Limited programs included	Many programs prepacked + package management systems
Programs (at once)	Multitasking	Multitasking
Kernel	Hybrid	Monolithic
User Design Philosophy	User-friendly	Empower users but doesn't hold their hands

Brief UNIX History

The Unix Timeline

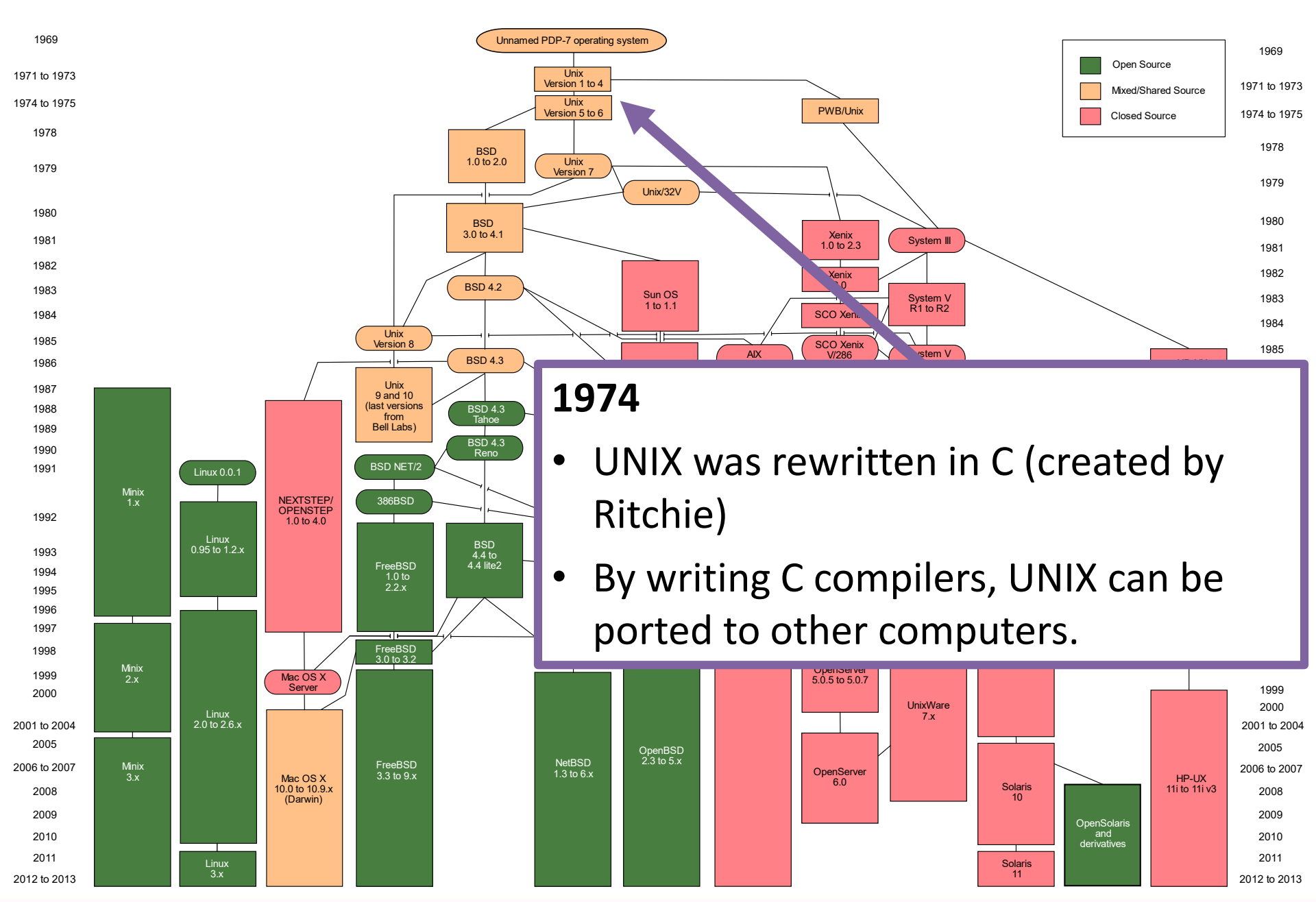
Site: https://www.levenez.com/unix/


Open Source
Mixed/Shared Source
Closed Source
1969
1971 to 1973
1974 to 1975
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001 to 2004
2005
2006 to 2007
2008
2009
2010
2011
2012 to 2013
Unnamed PDP-7 operating system
Unix Version 1 to 4
Unix Version 5 to 6
Unix Version 7
PWB/Unix
BSD 1.0 to 2.0
Unix/32V
BSD 3.0 to 4.1
BSD 4.2
BSD 4.3
Unix Version 8
Unix 9 and 10 (last versions from Bell Labs)
BSD 4.3 Tahoe
BSD 4.3 Reno
BSD NET/2
386BSD
FreeBSD 1.0 to 2.2.x
FreeBSD 3.0 to 3.2
FreeBSD 3.3 to 9.x
BSD 4.4 to 4.4 lite2
NetBSD 0.8 to 1.0
NetBSD 1.1 to 1.2
NetBSD 1.3
NetBSD 1.3 to 6.x
OpenBSD 1.0 to 2.2
OpenBSD 2.3 to 5.x
Sun OS 1 to 1.1
Sun OS 1.2 to 3.0
Sun OS 4
AIX 1.0
AIX 3.x to 7.1
Xenix 1.0 to 2.3
Xenix 3.0
SCO Xenix
SCO Xenix V/286
SCO Xenix V/386
SCO Unix 3.2.4
OpenServer 5.0 to 5.04
OpenServer 5.0.5 to 5.0.7
OpenServer 6.0
System III
System V R1 to R2
System V R3
System V R4
UnixWare 1.x to 2.x
UnixWare 7.x
Solaris 2.1 to 9
Solaris 10
Solaris 11
OpenSolaris and derivatives
HP-UX 1.0 to 1.2
HP-UX 2.0 to 3.0
HP-UX 6 to 11
HP-UX 11i to 11i v3
Minix 1.x
Minix 2.x
Minix 3.x
Linux 0.0.1
Linux 0.95 to 1.2.x
Linux 2.0 to 2.6.x
Linux 3.x
NEXTSTEP/ OPENSTEP 1.0 to 4.0
Mac OS X Server
Mac OS X 10.0 to 10.9.x (Darwin)



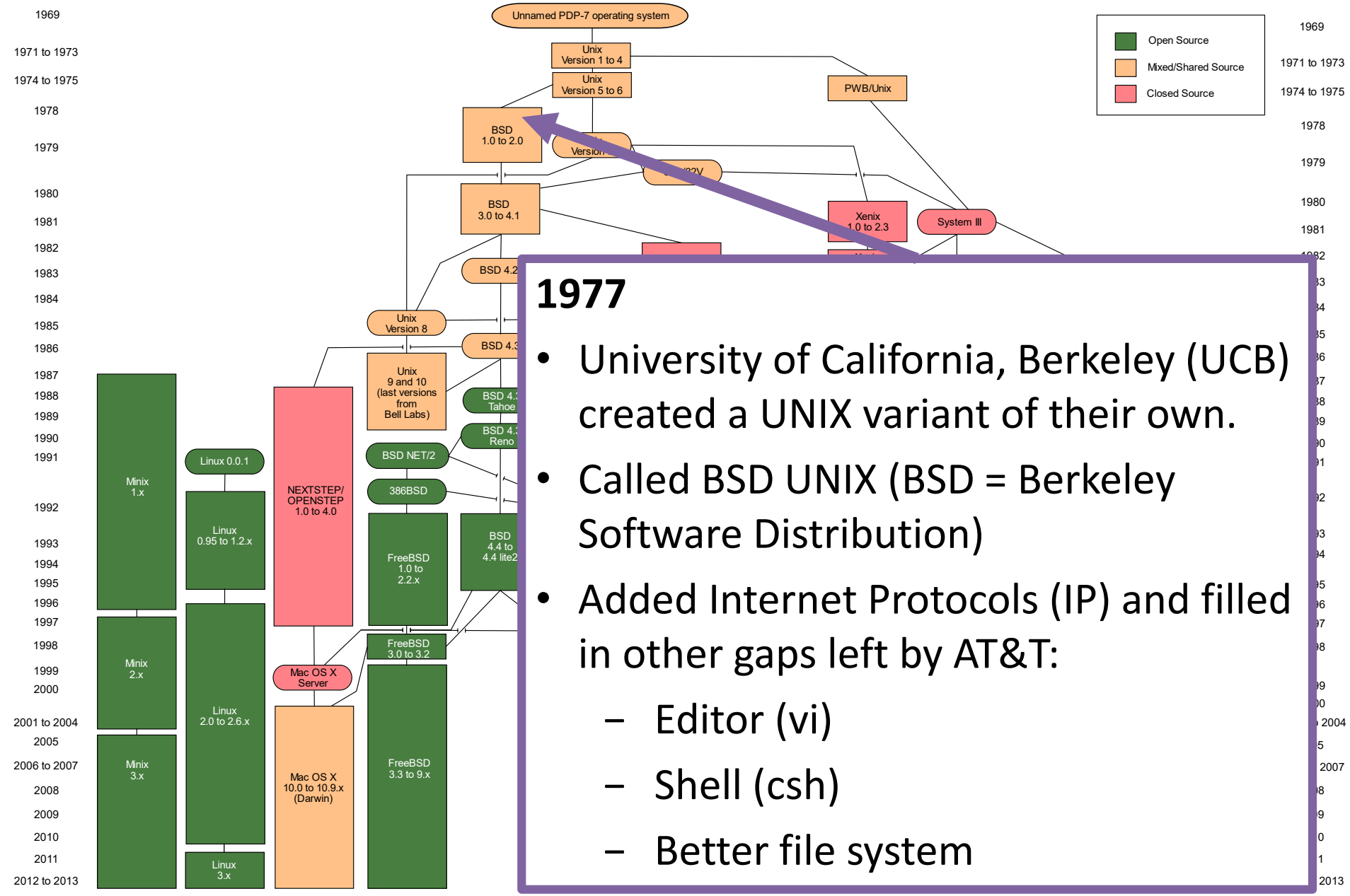
1969

- Ken Thompson and Dennis Ritchie of AT&T labs author the first UNIX system
- Written in assembly language for one specific machine



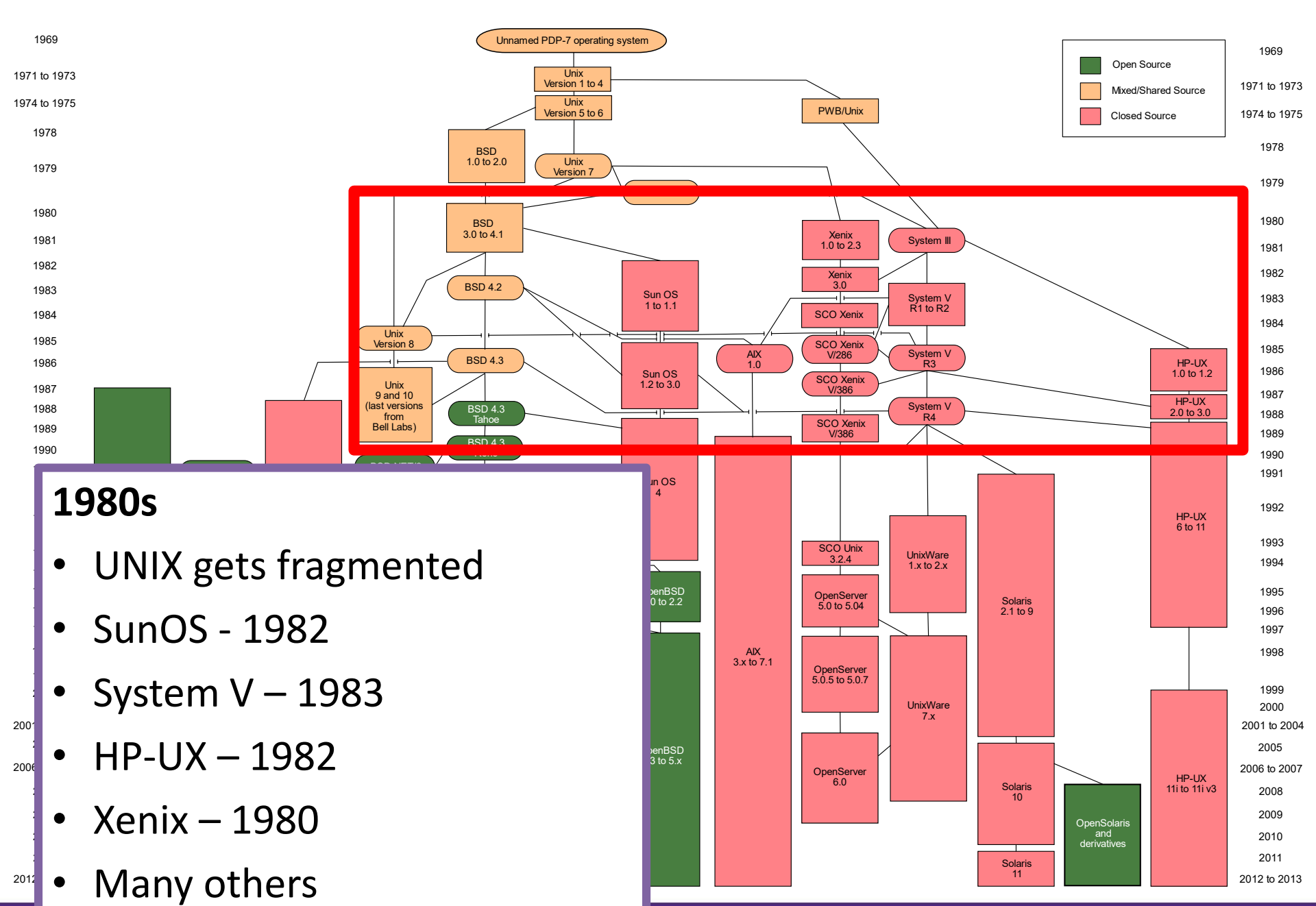
1974

- UNIX was rewritten in C (created by Ritchie)
- By writing C compilers, UNIX can be ported to other computers.



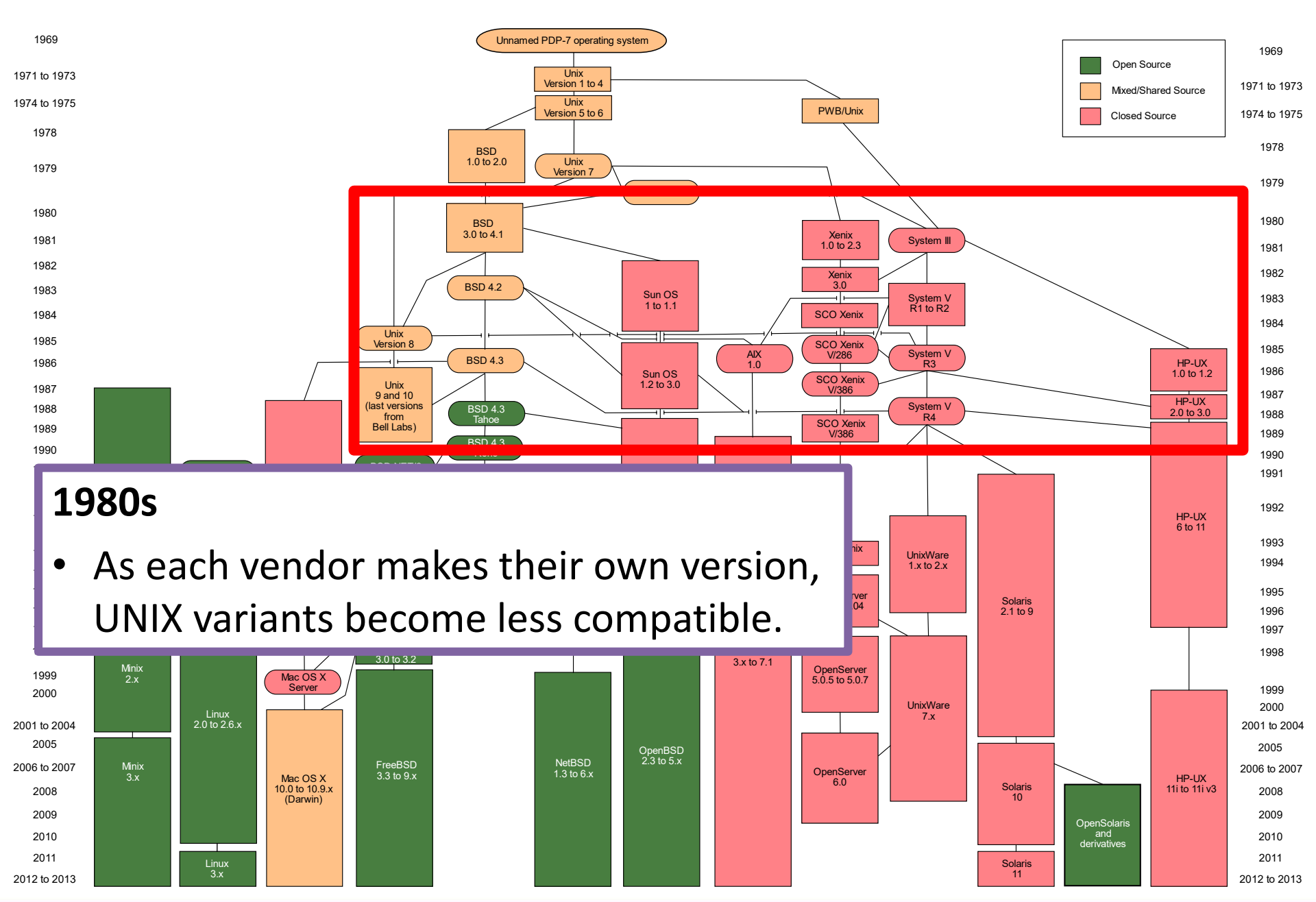
1977

- University of California, Berkeley (UCB) created a UNIX variant of their own.
- Called BSD UNIX (BSD = Berkeley Software Distribution)
- Added Internet Protocols (IP) and filled in other gaps left by AT&T:
  - Editor (vi)
  - Shell (csh)
  - Better file system



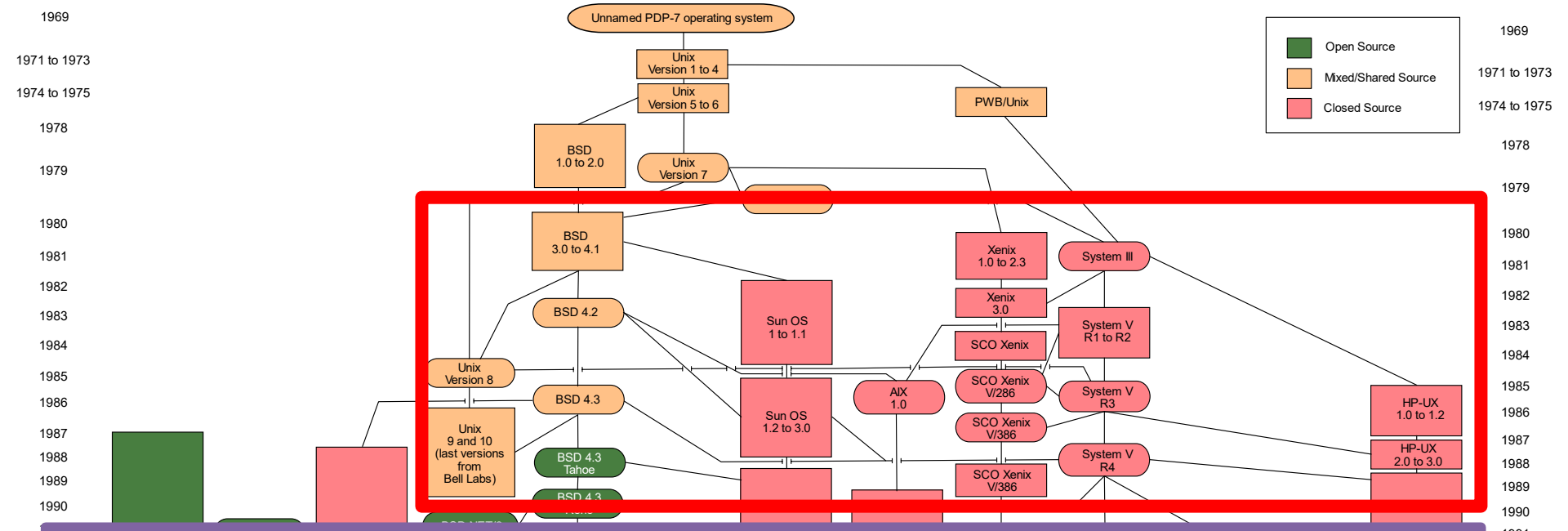
1980s

- UNIX gets fragmented
- SunOS - 1982
- System V – 1983
- HP-UX – 1982
- Xenix – 1980
- Many others



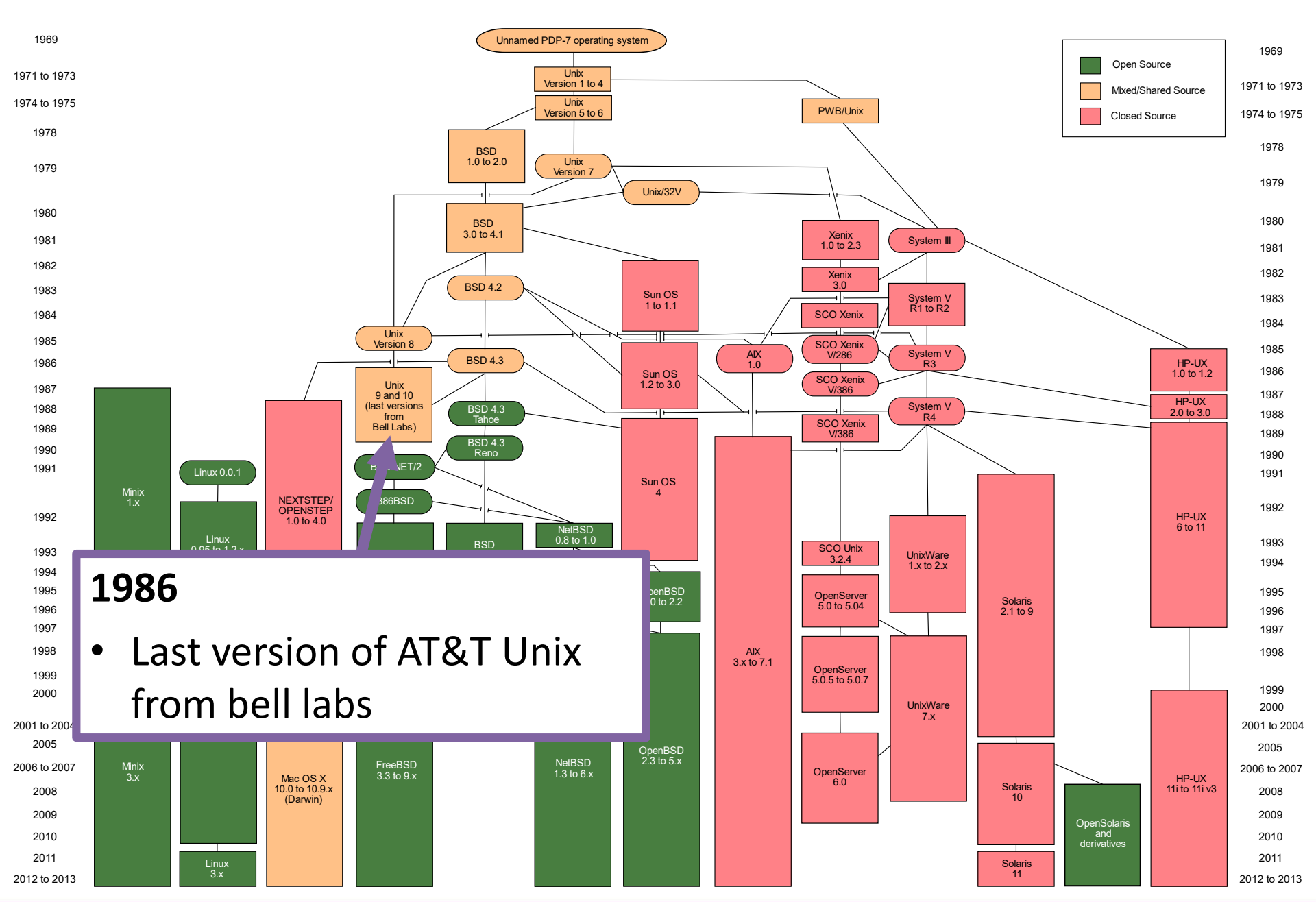
1980s

- As each vendor makes their own version, UNIX variants become less compatible.



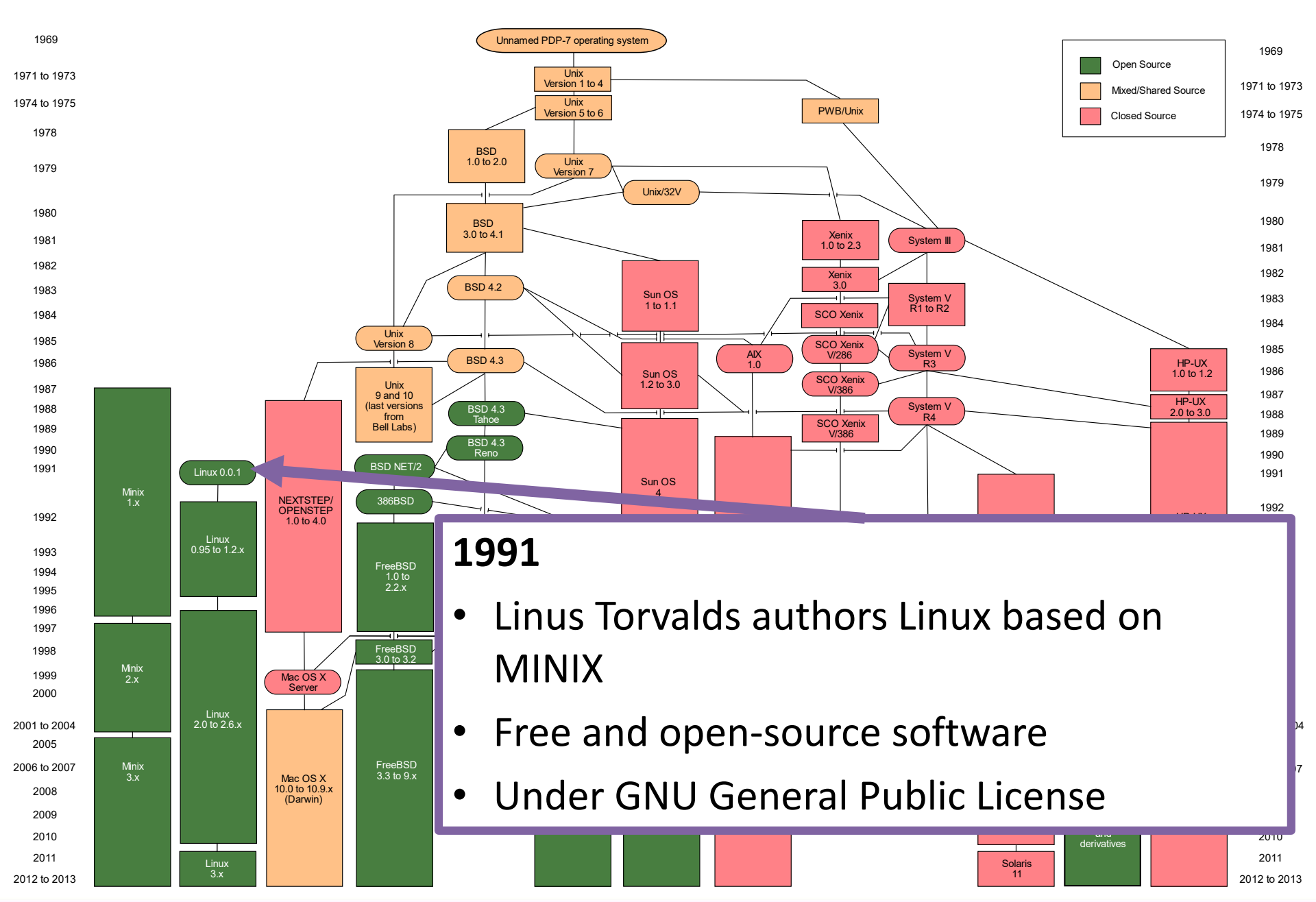
1984

- The Portable Operating System Interface (POSIX) is created to standardize OSs and maintain compatibility
- Standardizes operations, command line utilities, environment variables, shell languages, regular expression, directory structure, filenames and more
- Most UNIX-like OSs follow POSIX



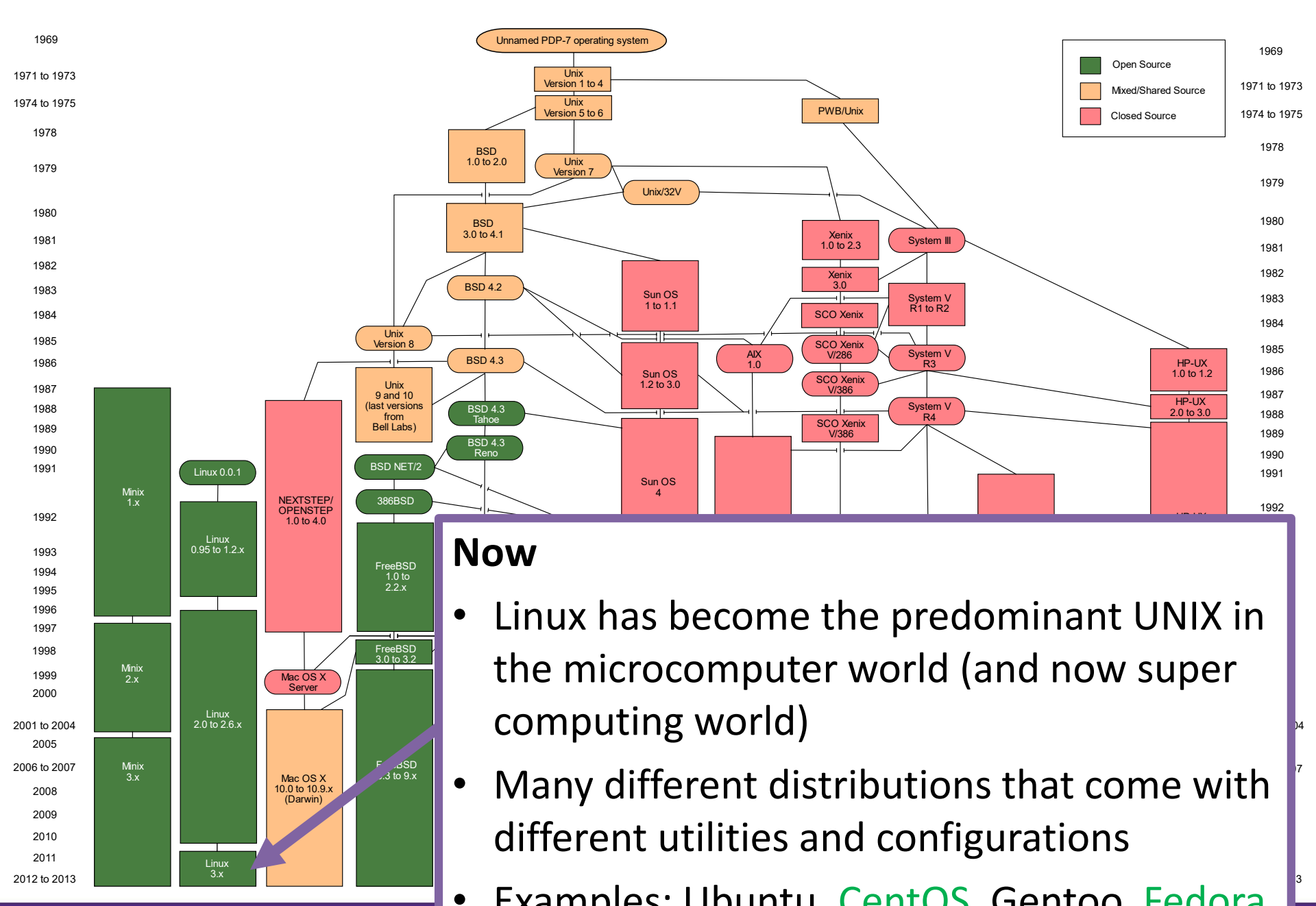
1986

- Last version of AT&T Unix from bell labs



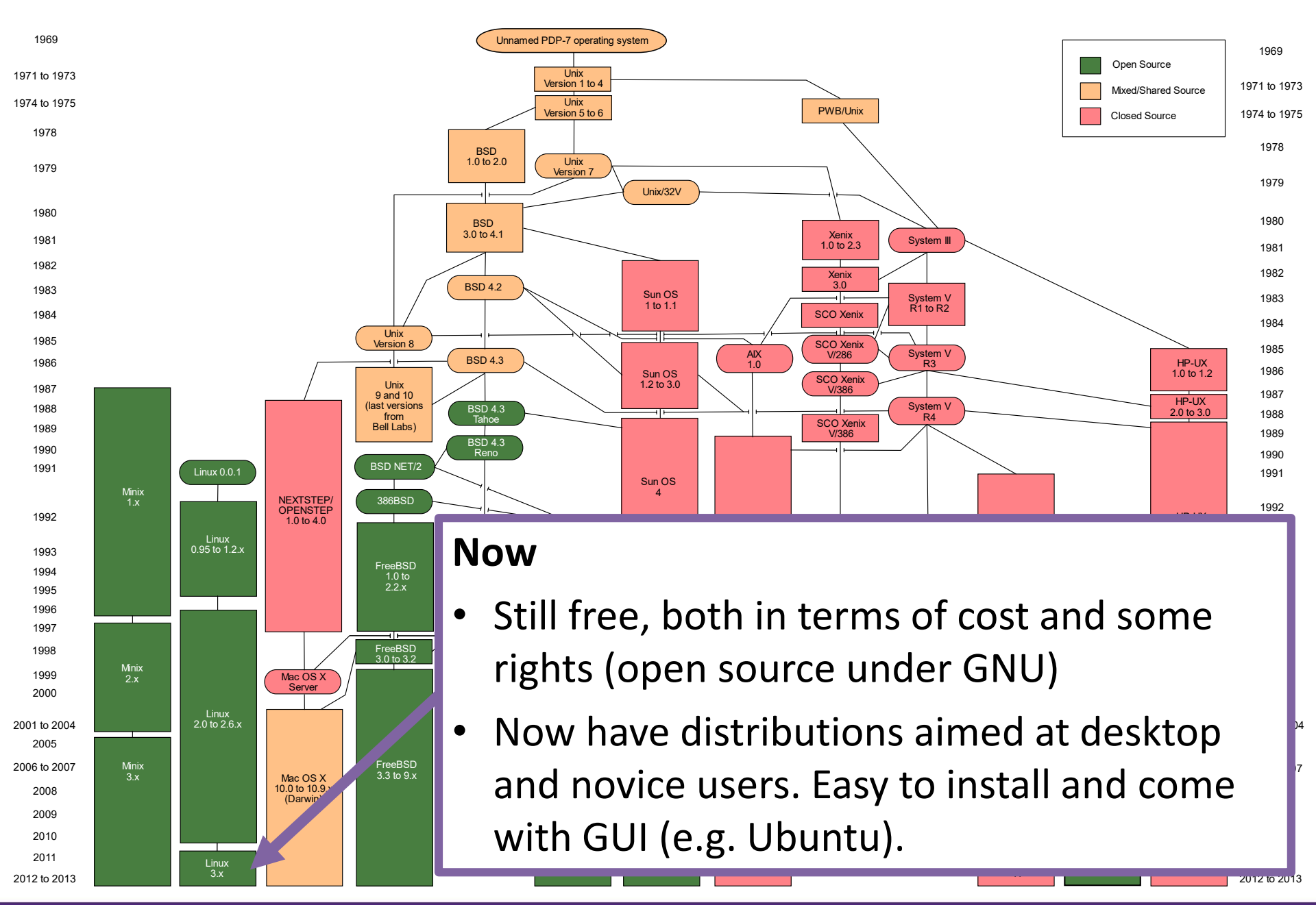
1991

- Linus Torvalds authors Linux based on MINIX
- Free and open-source software
- Under GNU General Public License



Now

- Linux has become the predominant UNIX in the microcomputer world (and now super computing world)
- Many different distributions that come with different utilities and configurations
- Examples: Ubuntu, CentOS, Gentoo, Fedora



Now

- Still free, both in terms of cost and some rights (open source under GNU)
- Now have distributions aimed at desktop and novice users. Easy to install and come with GUI (e.g. Ubuntu).

Operating Systems we Will be Using

Course Server (cs2211b.gaul.csd.uwo.ca)

- CentOS Linux

MC244 Computers

- Fedora Linux

Obelix (obelix.gaul.csd.uwo.ca)

- Solaris (SunOS)
- Only in one lab

The UNIX Philosophy

- Unix was not meant to be a user-friendly operating system
  - Instead, meant to be “user-helpful” and very powerful
- The Unix operating system protects users from other users but not necessarily from themselves
- It provides the necessary tools then gets out of your way

The UNIX Philosophy

UNIX Tools

- Keep each tool simple
- Have each tool do one thing, and do that one thing really well
- Keep tools concise and not too talkative
- More complex tasks can be accomplished by combining tools together in scripts or pipelines
- Originally, input and output to workstations were slow and tedious, and this approach made things faster and more efficient.

UNIX/Linux Basics

Users

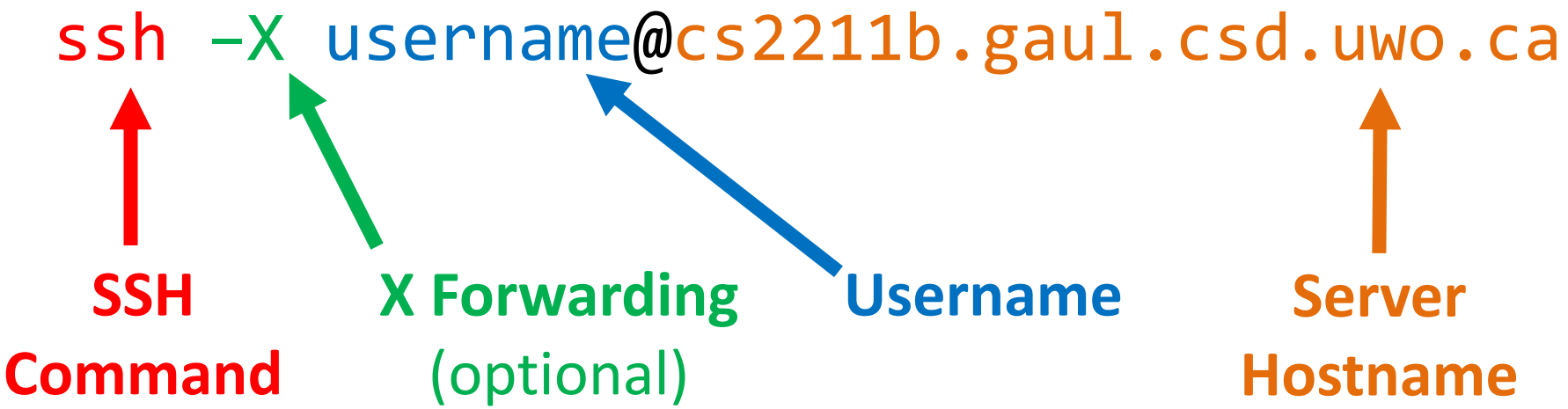
- Must have an account to use a UNIX computer
  - To share resources, need to tell users apart
  - To restrict access to only those we want on our server
- You can only access the resources that are specified by your account information
  - Accounts track, control, and limit user activity
- There is at least one super user account in a system usually named “root”, who has absolute power over the system (On some Windows systems, this account is named “administrator”)

SSH

- Log in remotely using secure shell (SSH)
- SSH helps emulate a terminal like you were sitting at a computer with no GUI (shell only)
- You can log in from anywhere with an internet connection!
- Instructions on OWL
  - Assigned Readings → Week 1 → How to Connect to the Course Server
- Will be covered in the first lab if you are having issues right now

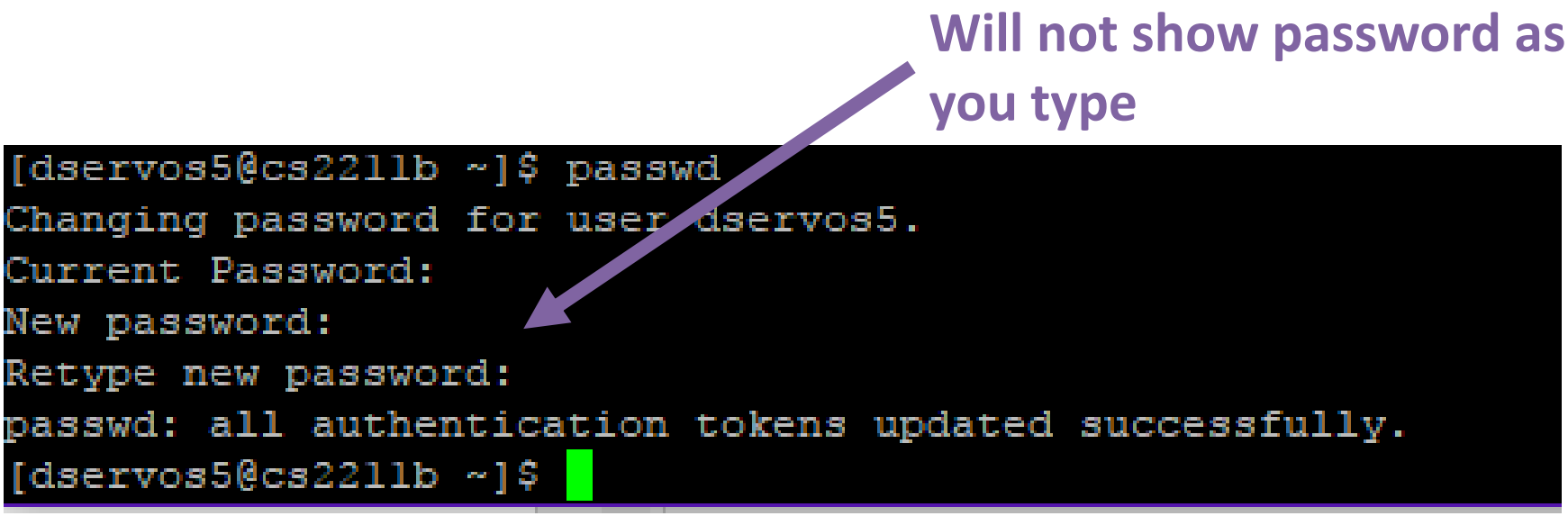
SSH

On UNIX/Linux the command looks like this:



Change Your Password!

First thing you should do is change your password using the `passwd` command:



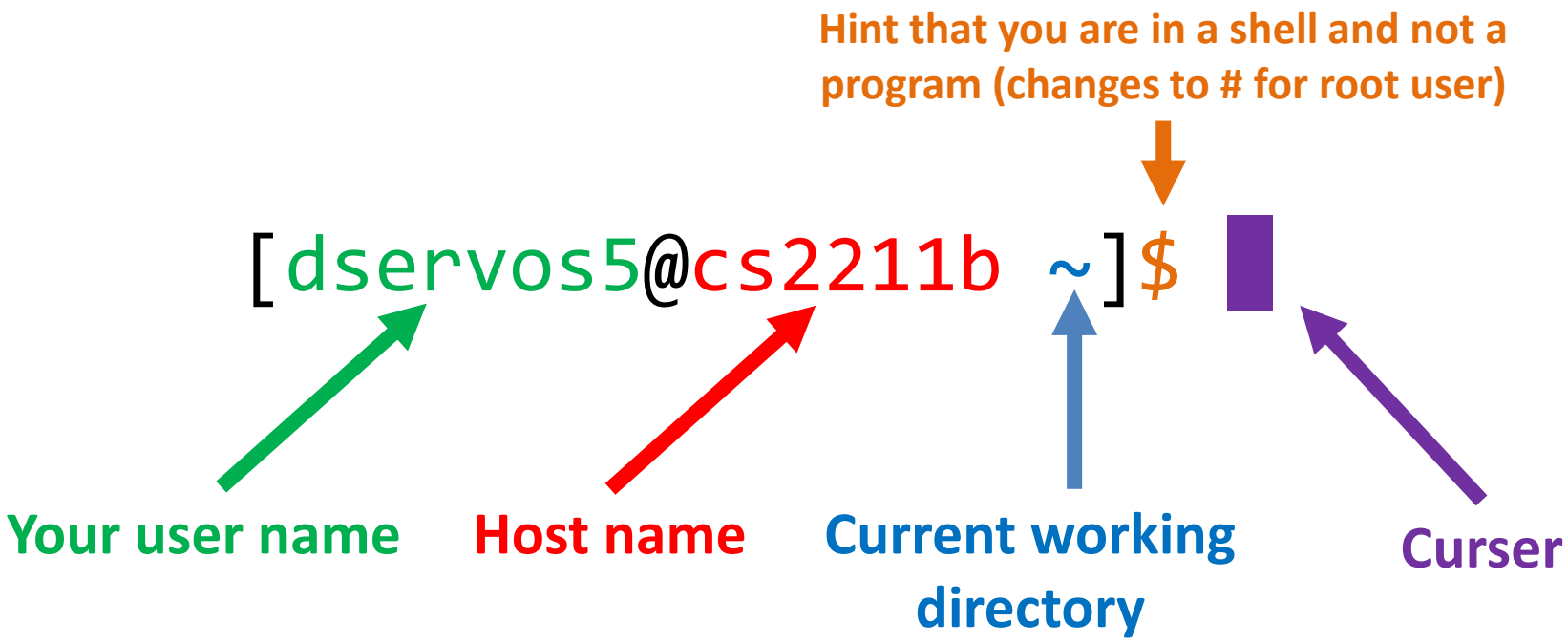
The Shell

- Default shell is Bash (Bourne-again shell)
- Will show prompt at first that looks like this:

```
[dservos5@cs2211b ~]$
```

The Shell

- Default shell is Bash (Bourne-again shell)
- Will show prompt at first that looks like this:



The Shell

- Allows you to type and execute text based commands that are in your PATH
- For example:
  - `who`
  - `finger`
  - `date`
  - `hostname`
  - `cal`
  - `pwd`

Command Structure

- Commands are made up of several parts
  - Command
  - Arguments
  - Options
  - Option Arguments

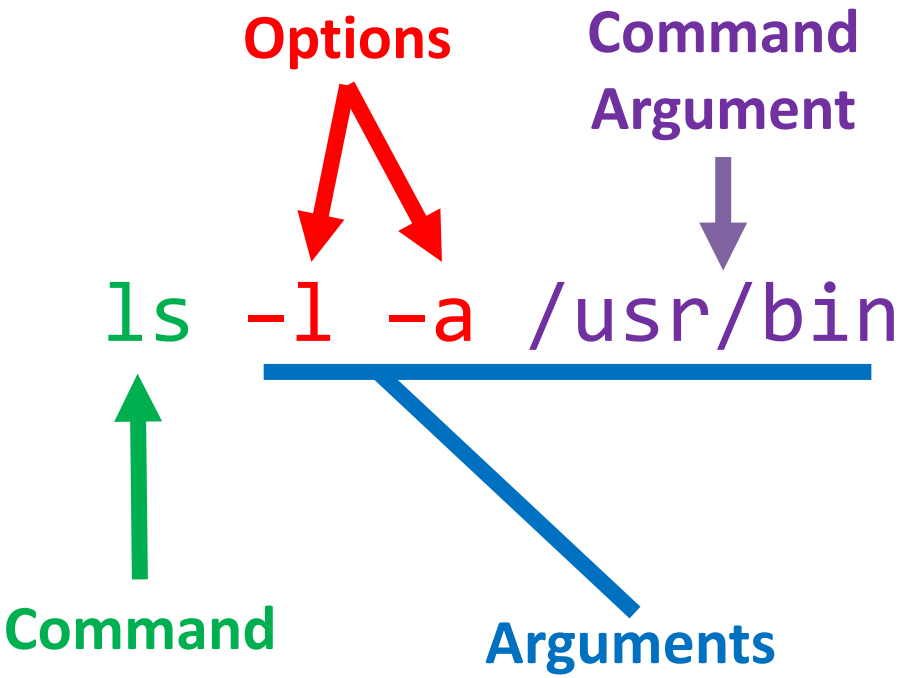
Command Structure

Example 1:

```
ls -l -a /usr/bin
```

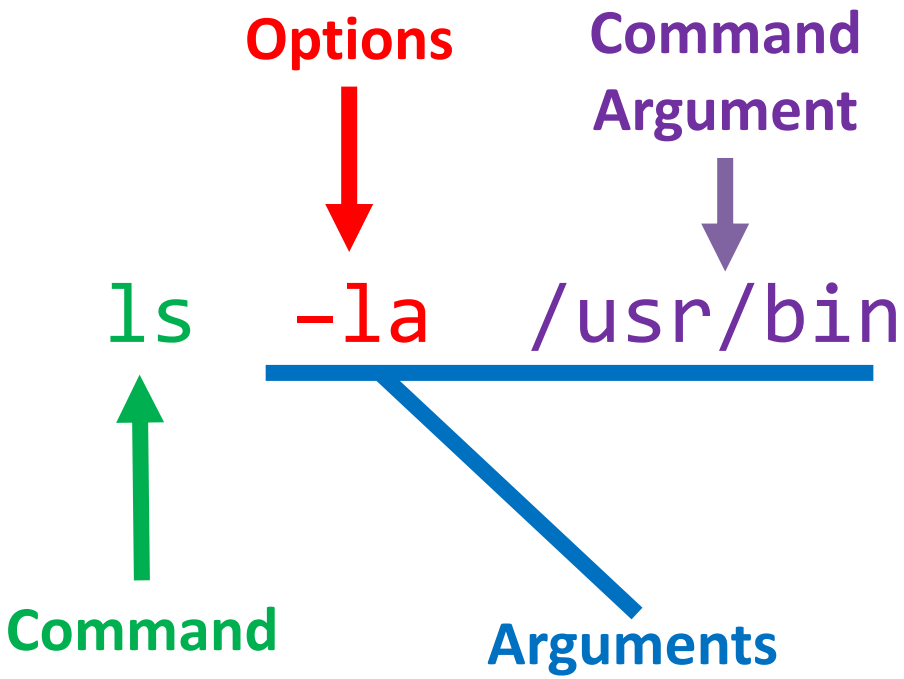
Command Structure

Example 1:



Command Structure

Equivalent:



Options can often be combined with one -

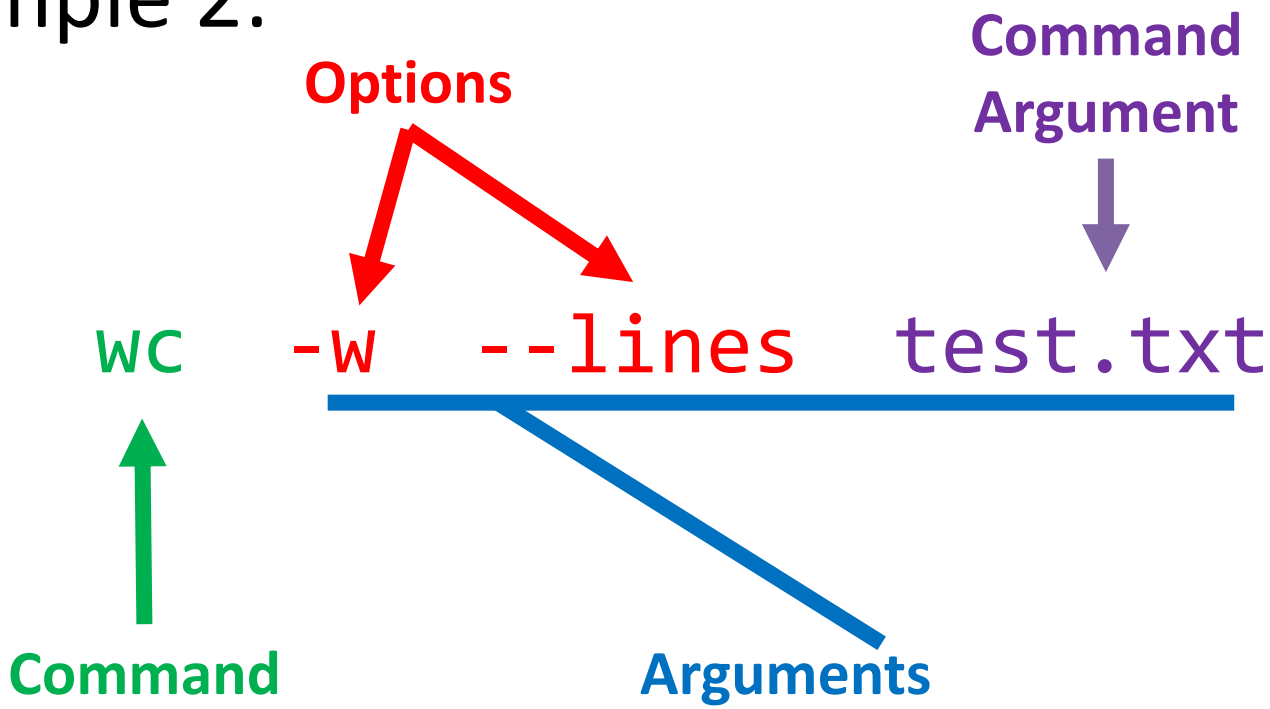
Command Structure

Example 2:

```
wc  -w  --lines  test.txt
```

Command Structure

Example 2:



Command Structure

Equivalent:

```
wc  --words  -l  test.txt
wc  -w  -l  test.txt
wc  --words  --lines  test.txt
```

Some options have a long form with two --s

Command Structure

Cannot combine long form options like this:

```
wc  --wordslines  test.txt
```

```
[dservos5@cs2211b ~]$ wc --wordslines test.txt
wc: unrecognized option '--wordslines'
Try 'wc --help' for more information.
```

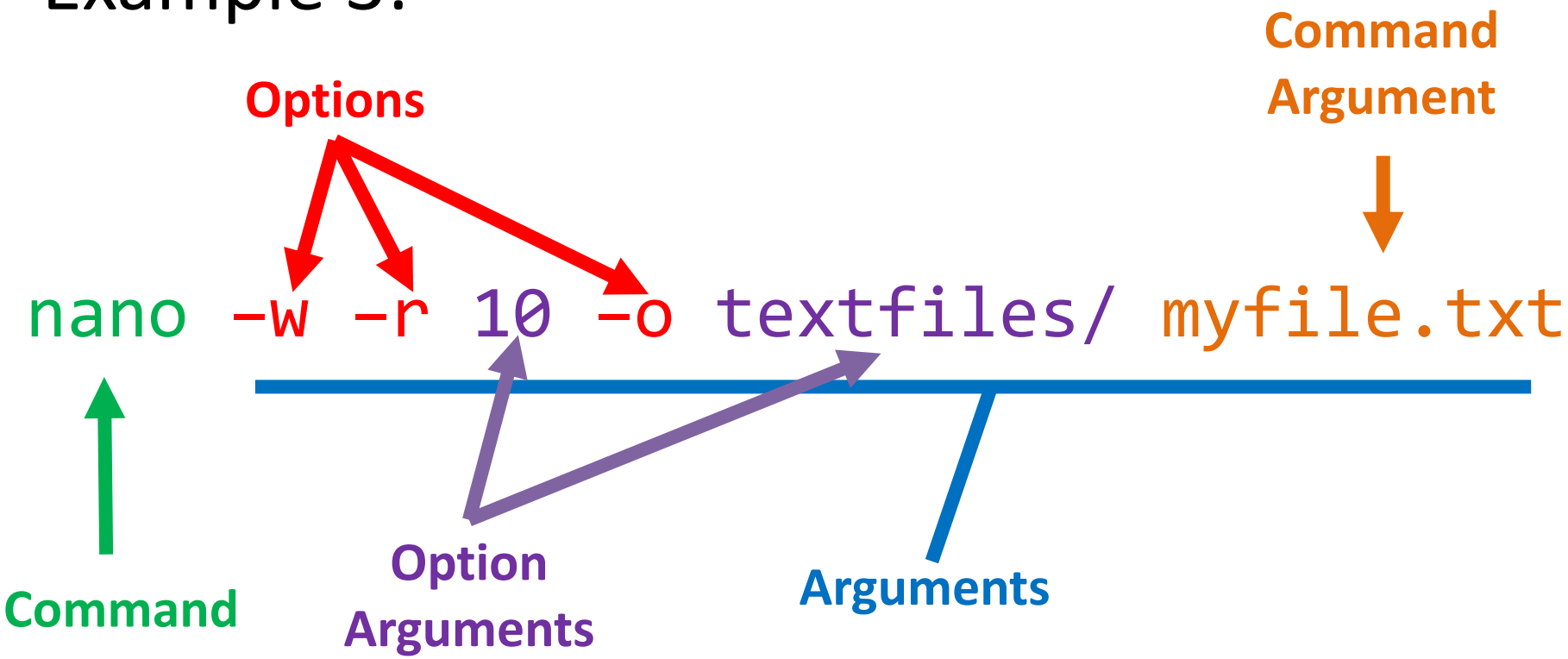
Command Structure

Example 3:

```
nano –w –r 10 –o textfiles/ myfile.txt
```

Command Structure

Example 3:



In-class Activity

In groups of 4 or 5 identify the parts of these commands as Command, Command Arguments, Options, or Option Arguments

1. `echo hello world!`
2. `ls –lad file1.txt file2.sh`
3. `nano --const –r 25 log.txt`

In-class Activity

Command	Options	Option Arguments	Command Arguments
`echo`			`hello world!`

`ls`	`l`
	`a`
	`d`
			`file1.txt`
			`file2.sh`

`nano`	`const r`	`25 (for r)`	`log.txt`

Getting Help

- Manual available for most commands
- Can be access via the command line:

```
man command_name
```

- Example, manual page for `who`:

```
man who
```

Getting Help

- Output:

```
WHO(1)

NAME
       who - show who is logged on

SYNOPSIS
       who [OPTION]... [ FILE | ARG1 ARG2 ]

DESCRIPTION
       Print information about users who are currently logged in.

       -a, --all
              same as -b -d --login -p -r -t -T -u

       -b, --boot
              time of last system boot

       -d, --dead
              print dead processes

       -H, --heading
              print line of column headings

       -l, --login
              print system login processes
```

Getting Help

- Navigate manual page using:
  - arrow keys (move up and down page)
  - e and y (also move up and down)
  - f and b (forward and backward by one window)
  - q (exit man)
- Many commands also have help options accessible with `--help` or `-h`
- Not always supported

Getting Help

- Example: `wc --help`
- Output:

```
[dservos5@cs221lb ~]$ wc --help
Usage: wc [OPTION]... [FILE]...
  or:  wc [OPTION]... --files0-from=F
Print newline, word, and byte counts for each FILE, and a total line if
more than one FILE is specified.  With no FILE, or when FILE is -,
read standard input.  A word is a non-zero-length sequence of characters
delimited by white space.
The options below may be used to select which counts are printed, always in
the following order: newline, word, character, byte, maximum line length.
  -c, --bytes            print the byte counts
  -m, --chars            print the character counts
  -l, --lines            print the newline counts
      --files0-from=F    read input from the files specified by
                           NUL-terminated names in file F;
                           If F is - then read names from standard input
  -L, --max-line-length  print the length of the longest line
  -w, --words            print the word counts
      --help     display this help and exit
      --version  output version information and exit

GNU coreutils online help: <http://www.gnu.org/software/coreutils/>
For complete documentation, run: info coreutils 'wc invocation'
```

Getting Help

- Can always Google it!
- Many man pages and other resources are available online.
- Web based man pages:
  - http://man.he.net
  - https://linux.die.net/man
  - https://man.cx
  - Many others

More Basic Commands

Command	Description
`ls`	Lists files in a given directory. Uses the current directory by default
`pwd`	Lists the current working directory (the directory that you are in)
`who`	Displays the current users logged into the system
`finger`	Displays detailed information on users currently logged in or on a specified user
`date`	Display the current date and time. Can give an argument to change the format of the date
`cal`	Display a calendar. For the current moth by default
`echo`	Print the given arguments to the screen

More Basic Commands

Command	Description
`cat`	Output a file to the screen (also outputs the given stdin)
`man`	Display the manual page for a given command.
`logout`	Exit the shell and logoff the server
`exit`	Same as logout
`tail`	Display the last X lines of a file
`head`	Display the first X lines of a file
`less`	Allows you to scroll through a file using arrow keys (q to quit)

More Basic Commands

Command	Description
`cd`	Change your current directory
`mv`	Move a file or directory
`rm`	Delete (remove) a file
`rmdir`	Delete (remove) a directory
`nano`	Text editor (same as pico on server)
`type`	Indicates type of command (built into shell or executable)
`whereis`	Gives location of command and it's man pages
`which`	If multiple copies of a command exit, tells you which one will be used

More Basic Commands

Command	Description
`wc`	Count words, lines, bytes, etc. in a file (or from stdin)
`printf`	Format and print arguments to the screen
`passwd`	Change your password
`ssh`	Create a secure shell connection to another system
`ps`	Reports current running processes (just yours by default)
`ln`	Make a link between files.
`vi`	A more advanced but harder to use text editor than nano or pico.

More Basic Commands

- Recommend that you try each command out your self on the course server
- Can check man pages to see how they work
- Will cover some of these in the labs and future lectures