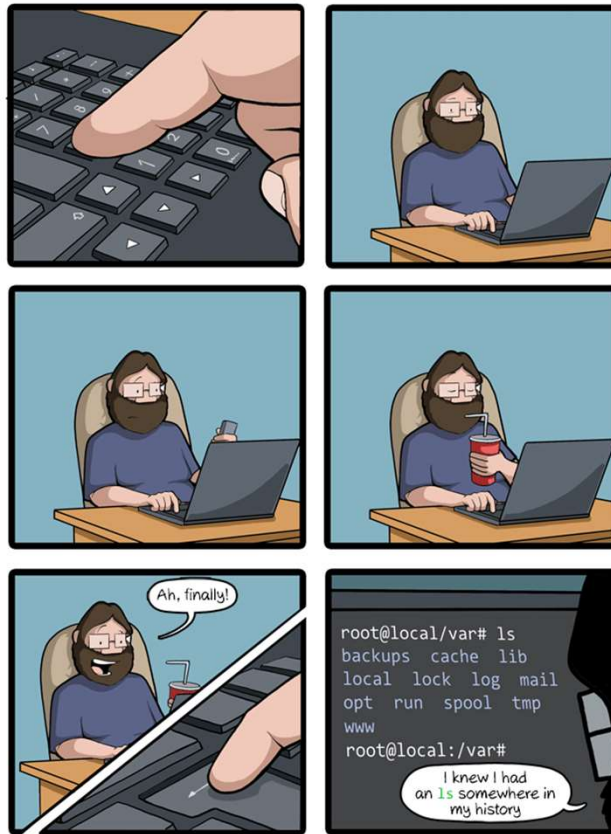




Western
UNIVERSITY • CANADA

Shell environments

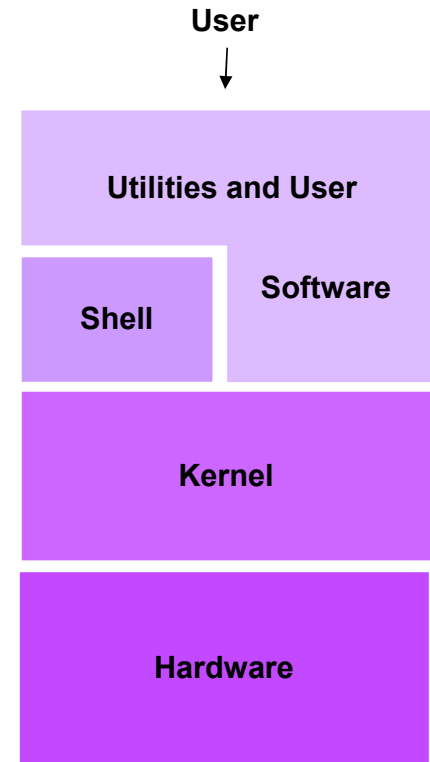
Winter 2022



CommitStrip.com

Introduction to Unix

- Parts of a Unix operating system
 - Utilities – Standard tools and applications
 - Shell – An interface between users and the kernel. Provides a command-line interpreter
 - Kernel – Manages the processes, resources, hardware



The shell

- We've spent most of our time focusing on Unix utilities (vi, grep, pipelines, etc.)
- Let's focus on the shell
 - The shell is interactive software that exposes the operating system functions to the user (the outer "shell" of the OS)
 - They can be graphical user (GUI) or command-line (CLI) interfaces

The shell

- Unix has a history of shells
 - `/bin/sh` – "Thompson shell" was the first shell developed by Ken Thompson in 1971. It notably lacked serious programming features
 - `/bin/sh` – "Bourne shell" developed by Stephen Bourne in 1979 replaced the Thompson shell and is still the most basic shell installed on all Unix systems

The shell

- Unix has a history of shells
 - `/bin/csh` – "C shell" was the first shell developed by Bill Joy in 1978. The syntax was more "C-like"
 - `/bin/tcsh` – An improved "C shell" and largely superseded it

The shell

- Unix has a history of shells
 - `/bin/ksh` – "Korn shell" was the first shell developed by David Korn in the early 1980s. It is backwards compatible with the Bourne shell and includes features from the "C shell"

The shell

- Unix has a history of shells
 - `/bin/zsh` – Developed in the 1990s and combines most features of the "Bourne shell", "C shell", the "Korn shell", and Bash
 - Notably, zsh is now the default on MacOS and several Linux distros

The Bourne Again Shell

- Unix has a history of shells
 - `/bin/bash` – First developed in 1989 and combines most features of the "Bourne shell", "C shell", the "Korn shell"
 - This is by far the most popular shell and the default on Gaul so this will be our focus

The Bourne Again Shell

- Your username has been setup on Gaul to use the Bash shell. When you first log in, you get the Bash shell.

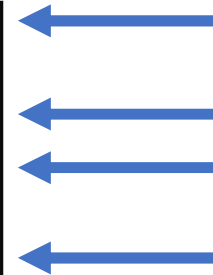
```
[wbeldman@compute ~]$ getent passwd wbeldman  
wbeldman:*:2001066:2001066:William Beldman:/home/wbeldman:/bin/bash
```



The Bourne Again Shell

- The other shells mentioned are installed and you could use them if you wish

```
[wbeldman@compute ~]$ echo $0  
-bash  
[wbeldman@compute ~]$ /bin/zsh  
[wbeldman@compute]~% echo $0  
/bin/zsh  
[wbeldman@compute]~% exit  
[wbeldman@compute ~]$ _
```



Shell variables

- Your shell is highly dependent on what your shell variables are set to
- There are two types of variables
 - Environmental variables – Variables that are available to the shell and any programs you run
 - Non-environmental variables – Variables that are available to the shell only

Shell variables

- Environmental variables
 - To see these variables, run the `export` or `env` command without arguments
 - To set these variables, run
`export VAR_NAME=VALUE`
 - By convention, we use upper case letters and underscores since spaces are not allowed in variable names

Shell variables

- Non-environmental variables
 - It is not easy to see just these variables, but they are usually only used in scripts
 - To set these variables, simply run
`var_name=VALUE`
 - By convention, we use lower case letters and underscores since spaces are not allowed in variable names

Shell variables

- You have a number of environment variables already set
- These variables were set automatically when you logged in via `/etc/bashrc` or `~/ .bashrc`
- You can set your own in `~/ .bashrc` (but be cautious when modifying this file)
- You can temporarily modify variables on the command-line

Shell variables

- To reference these variables, we use the \$ symbol

```
[wbeldman@compute ~]$ export ENVIRONMENTAL_VARIABLE=1
[wbeldman@compute ~]$ non_environmental_variable=1
[wbeldman@compute ~]$ echo $HOME
/home/wbeldman
[wbeldman@compute ~]$ echo $ENVIRONMENTAL_VARIABLE
1
[wbeldman@compute ~]$ echo $non_environmental_variable
1
[wbeldman@compute ~]$ _
```


Shell variables

- Important environmental variables to know
 - `$HOME` – The user's home directory
 - `$SHELL` – The user's shell
 - `$PATH` – Where to find commands
 - `$PWD` – The current working directory
(this is updated whenever you change directories)

Shell variables

- Important environmental variables to know
 - `$TERM` – Used by some utilities to know what kind of terminal window you have (and what characters your terminal supports)
 - `$HOSTNAME` – The name of the current system you are logged in to


\$PATH

- Important notes about the `$PATH` variable
- Bash uses this variable to determine where to find a command you are running
- It is a list of directories separated by the `:`
- Bash will run the first command it finds in the directory found in left-to-right order

\$PATH

- Use the `whereis <commandname>` to find all instances of `<commandname>` installed on the system (and the man pages)
- Use `which <commandname>` to determine which instance is the default
- (Or in other words, which instance was the first one found in the `$PATH` variable)

\$PATH



```
[wbeldman@compute ~]$ echo $PATH
/usr/local/tetex/bin:/opt/SUNWspro/bin:/usr/ucb:/usr/bin:/usr/ccs/bin:/usr/local/bin:/usr/openwin/bin:/usr/sbin:.
[wbeldman@compute ~]$ whereis passwd
passwd: /usr/bin/passwd /etc/passwd /usr/local/bin/passwd /usr/share/man/man1/passwd.1.gz /usr/share/man/man5/passwd.5.gz
[wbeldman@compute ~]$ which passwd
/usr/bin/passwd
[wbeldman@compute ~]$
```

Command history

- Use the history command to see a list of commands in your history
- Use the up/down arrow to navigate through the history
- The maximum number of entries in the history list is controlled by the `$HISTSIZE` and `$HISTFILESIZE` variable

Command history

- Use the `!` symbol to search the history
 - `!!` – Re-run the previous command again
 - `!str` – Re-run the most recent command that begins with `str`
 - `!n` – Re-run the command at number `n` in the history list

Command history

```
[wbeldman@compute ~]$ history | tail -5
4143 history | tail -5
4144 cd /
4145 cd ~/
4146 pwd
4147 history | tail -5
[wbeldman@compute ~]$ !!
history | tail -5
4143 history | tail -5
4144 cd /
4145 cd ~/
4146 pwd
4147 history | tail -5
[wbeldman@compute ~]$ !pw
pwd
/home/wbeldman
[wbeldman@compute ~]$ !4144
cd /
[wbeldman@compute /]$ _
```


The tab key

- Use the Tab key to auto-complete
- Type the first few characters and press Tab to ask the shell to auto-complete
- If there is more than one possibility, press Tab twice to list all the choices



Western
UNIVERSITY • CANADA