OSU Oregon State University

CENTER FOR GENOME RESEARCH & BIOCOMPUTING

“Introduction to Unix/Linux”
INX_U18, Day 5, 2018-08-03

Installing programs, edit $path, .cshrc, source vs. binary, wget, tar, gzip, make

Learning Outcome(s):

Install and run software from your home directory.

Purpose of the $path shell variable and determine a program's location.

Download bioinformatics analysis programs and decompress them.

Matthew Peterson, OSU CGRB, matthew@cgrb.oregonstate.edu

Please do not redistribute outside of OSU; contains copyrighted materials.

"Installing" a program

- We have created two scripts, i.e., `myprog.sh` and `myprog.py`, and now want to run them from any directory or "install" them, i.e., run the from anywhere.

Steps so far:

1) Make sure it's executeable
2) Put it in a directory somewhere
3) Add the absolute path to that program's location to the `$path` (shell variable).

Oregon State UNIVERSITY

Step 2: "Storing" your programs

```
mkdir $HOME/local
mkdir $HOME/local/bin
```

- `$HOME/local/bin` is a "traditional" directory to store your own programs in.

```
mv myprog.sh $HOME/local/bin
mv myprog.py $HOME/local/bin
```

Oregon State UNIVERSITY

Step 3: Edit $path

```
set path = ("$HOME/local/bin" $path)
```

- We are assigning **$path** to a list values that includes the new absolute location, followed by the locations in the original **$path**
- **$path** a "shell variable" so we use **set** (instead of **setenv** for an "environment variable").

Oregon State UNIVERSITY

Bash: Edit $PATH

```
export PATH="$HOME/local/bin:$PATH"
```

- The bash shell (as opposed to tcsh) uses the **export** command (instead of **set**).
- The list of paths are separated by a **:**

Step 3: List order

```
set path = ("$HOME/local/bin" $path)
```

- This places our new directory at the start of the search path. You can also place it at the end:

```
set path = ($path "$HOME/local/bin")
```

- What could be the pros/cons of each?

Oregon State UNIVERSITY

Making this all "stick"

- If you logout you lose your new **`$path`**!
- You could re-login and re-**`set`** your **`$path`** (This would get tedious…)
- There's a way to have this happen automatically using specific "hidden" (dot files) in your **$HOME** directory:

`.login` and **`.cshrc`**

Oregon State UNIVERSITY

`.login` and `.cshrc`

- Commands in the `.login` file are run:
  when you login to your tcsh shell
- Commands in the **.cshrc** file are run:
  when you login **or** just start the tcsh shell
- "Login" implies entering your password (at the prompt)

Oregon State UNIVERSITY

.login vs. .cshrc

- **.cshrc** is a better place to put modifications to your **$path** as it will always be "**sourced**" (read) in your scripts, which start with **#!**/bin/tcsh
- Lets edit our **$path** :

```
nano –w $HOME/.cshrc
# Lines that start with a hash are comments
# added to include my /local/bin in $path
set path = ("$HOME/local/bin" $path)
```

- Logout/login, echo **$path**

Note / WARNING

- **Note: .login** and **.cshrc** may or may not exist depending on how the system administrator setup the system. If they do exist, change at your own risk.
- **WARNING**: Significant errors in these files could prevent you from logging in!

Oregon State UNIVERSITY

Installing Bioinformatics Software

Similar "recipe" for installation

1) Obtain executeable file(s)
2) Put them in `$HOME/local/bin`
3) Ensure `$HOME/local/bin` appears in **`$path`**

(we already did that!)

HMMER installation example

- HMMER searches for protein matches based on **H**idden **M**arkov **M**odels constructed from sets of similar protein sequences.
- The HMMER website used to offer both source code and binary versions for MacOSX and Windows.

- As of June, 2018 only source code is available.
- The textbook references to the site prior to June, 2018 offering both source and binary.

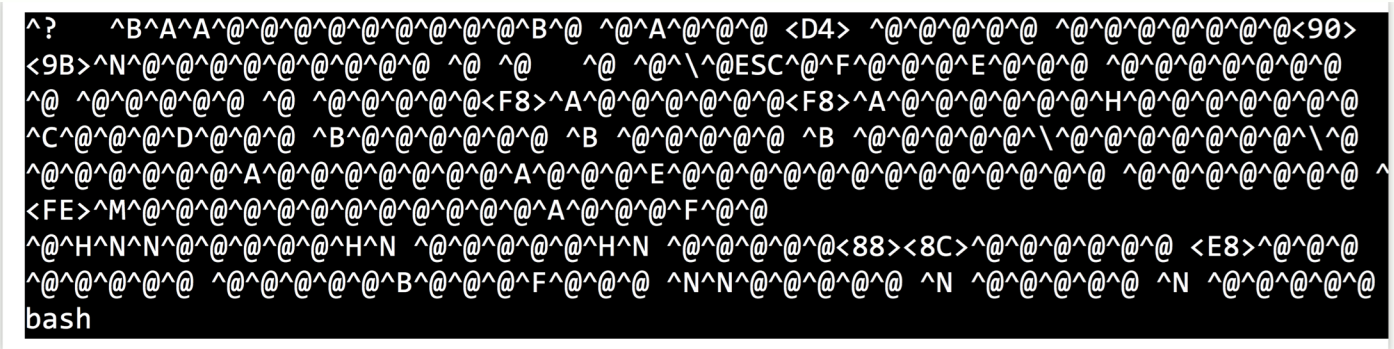
Oregon State UNIVERSITY

http://library.open.oregonstate.edu/computationalbiology/chapter/installing-bioinformatics-software/

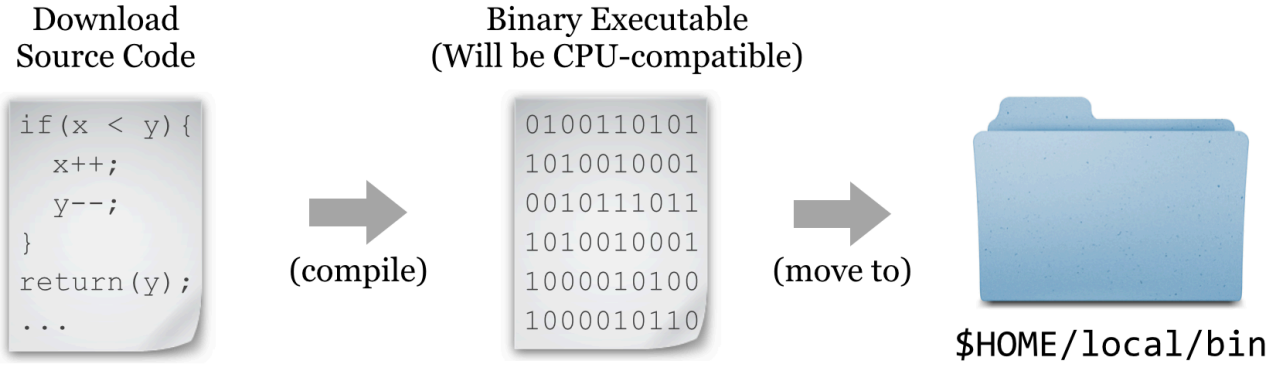
Source vs. Binary

- Some software is an interpreted (**source**) `text` script, e.g., `myprog.sh` that starts with **#!**
- A lot of software packages, e.g., `less`, are written in a compiled language, which implies the executeable is a **binary** file (1's and 0's) that runs directly on the CPU.

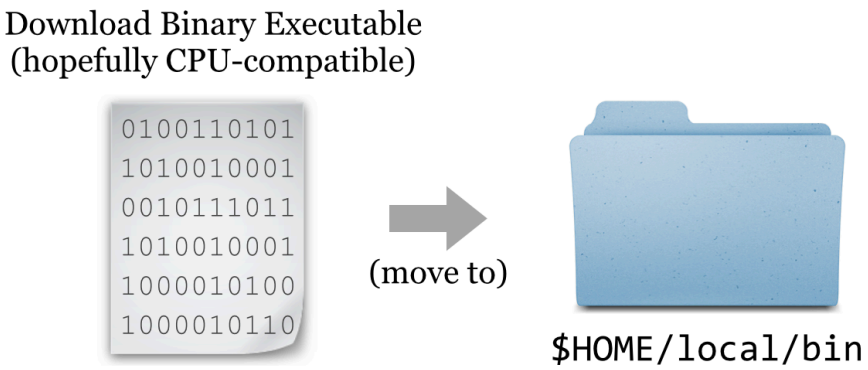
`less /bin/bash`



Download of Source vs. Binary



VS



http://teaching.cgrb.oregonstate.edu/CGRB/oneil/primer/partI_6_installing_bioinformatics_software.html

Pros/Cons of Source vs. Binary

Source

- Pros: Full control, more often available
- Cons: More difficult, may not work (missing dependencies)

Binary

- Pros: Easy
- Cons: May not be available! May not have some features enabled (uncommon)

Oregon State UNIVERSITY

Downloading the source for HMMER

- We want to download the **source** of HMMER to our **$HOME** directory on the server (not our Desktop).
- Lets create a downloads directory in **$HOME**

```
cd $HOME
mkdir downloads
cd downloads
```

- We are going to use a a program called **wget** to download the source.

Oregon State UNIVERSITY

`wget` HMMER source code

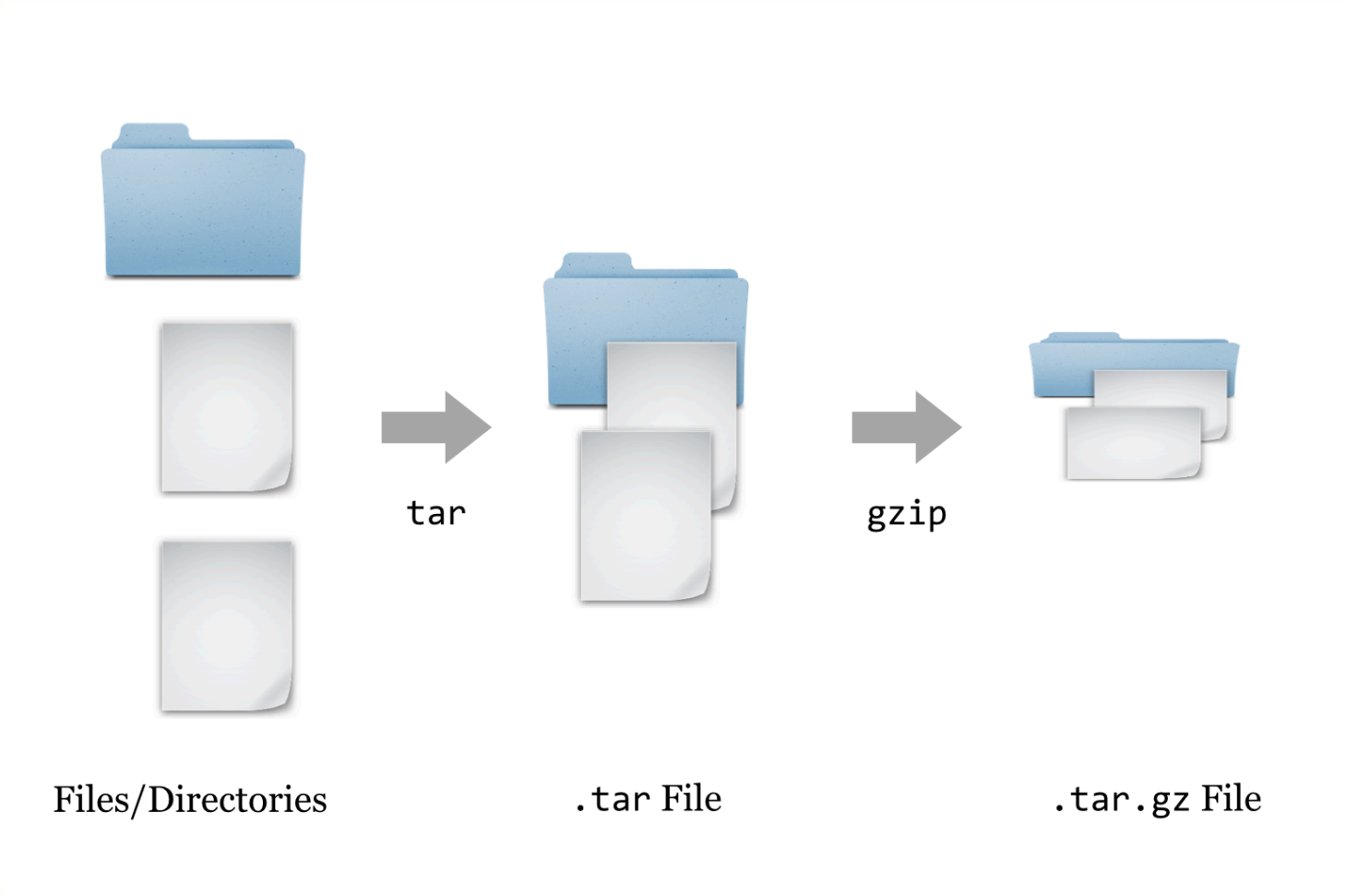
```
wget '<url>' -O <outputfilename>

# This command is all on one line
wget
'http://eddylab.org/software/hmmer/hmmer-3.2.1.tar.gz' -O hmmer-3.2.1.tar.gz
```

- `-O` is optional, may be necessary to keep filename
- We put the URL in quotes as it may contain *special* characters like `$`, `&`, and `?`

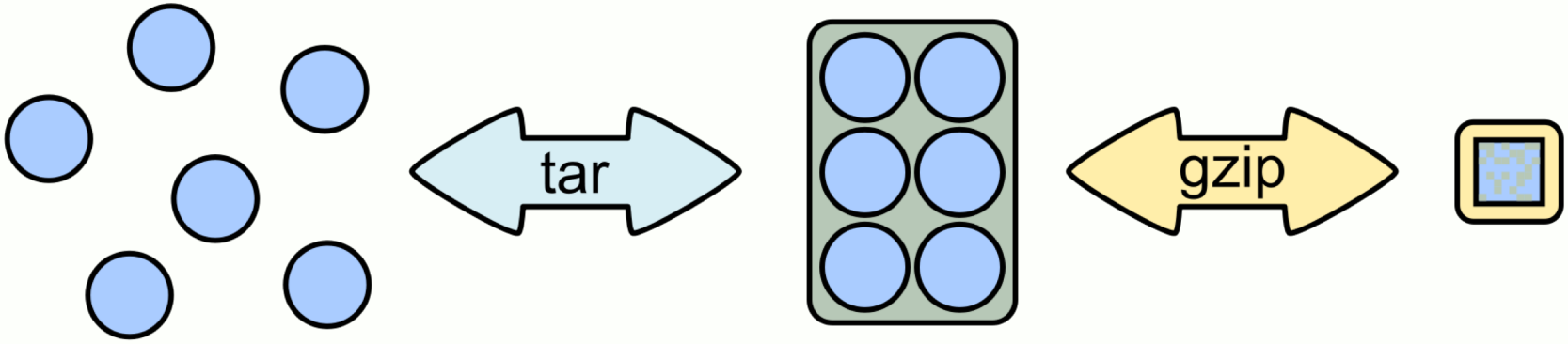
Oregon State University

Downloaded a `.tar.gz` file



http://library.open.oregonstate.edu/computationalbiology/chapter/installing-bioinformatics-software/

`.tar.gz` another look



https://en.wikipedia.org/wiki/Tar_(computing)

`gzip` Uncompressing a `.tar.gz`

```
gzip -d hmmer-3.2.1.tar.gz
gunzip hmmer-3.2.1.tar.gz
```

- To get the files "out" we need to uncompress:
  `hmmer-3.2.1.tar.gz`
- This will result in just the `.tar` file:
  `hmmer-3.2.1.tar`

`.gz` gone…

Oregon State UNIVERSITY

`tar` Extract the files in the `.tar`

```
tar -x -f <filename>
tar -xf hmmer-3.2.1.tar
```

- Take the single `.tar` file and extract the individual files and directories "inside" it using:

`-x` = "e**x**tract" and **`-f`** `<file`**`name`**`>`

- **`tar`** and **`gzip`** "care" about file extensions; most programs do not.

Oregon State UNIVERSITY

Other compression schemes

Extension	Decompress command
`.bz2`	`bunzip <target>`
`.zip`	`unzip <target>`
`.gz`	`gunzip <target>`
`.tgz`	`gunzip <target>`

`.tgz` is the same as as `.tar.gz`

Creating your own .tar.gz

```
tar -czf <tarballname> <dirname(s)>
tar -czf myhmmer.tar.gz hmmer-3.2.1
```

- You can create your own **.tar.gz** file (compressed tar file) from any directory

-c = "**c**reate"
-z = "g**z**ip"
-f <**f**ile**name**>

If you want to see it running, **-v** (**v**erbose)

Oregon State UNIVERSITY

`file` Identify a file (no extension)

```
file <filename>
```

```
cd downloads
mv hmmer-3.2.1.tar.gz test1234
file test1234
```

If you have a file that is missing an extension, e.g., **`.tar.gz`**, and you are unsure what type of file it is, the **`file`** command will attempt to identify it.

Oregon State UNIVERSITY

Compiling the source (code)

- We have downloaded the HMMER **source** code (primarily a set of text files).
- We need to turn this **source** code into a binary **ex**ecuteable (1's and 0's) that the server's CPU can directly run.
- Check for instructions! **README** or **INSTALL** file?

```
cd $HOME/downloads/hmmer-3.2.1
ls
less -S INSTALL
```

Oregon State UNIVERSITY

`./configure` and `make(s)`

- The INSTALL file shows four new commands:

```
./configure          # Configure
make                 # Build
make check           # Automated tests
make install         # Automated install
```

- `./configure`, `make`, and `make install` are part of the "canonical" install process on Unix/Linux
- `make check` is *optional* (tests the Build)
- `check` in some programs is instead `test`

Oregon State UNIVERSITY

./configure

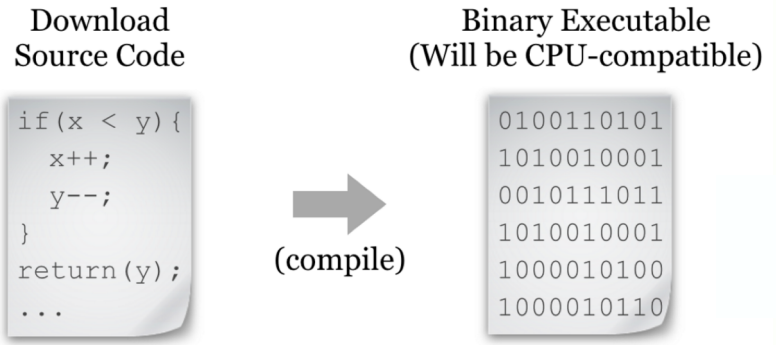
```
cd $HOME/downloads/hmmer-3.2.1
./configure
```

- An executeable script in your $PWD
- Checks to see if you have the right prerequisites to compile the program from **source** code
- Sets up environment variables in your shell
- May create (or alter) a **Makefile** that describes how the compile and install steps will work

Oregon State UNIVERSITY

make and Makefile

make

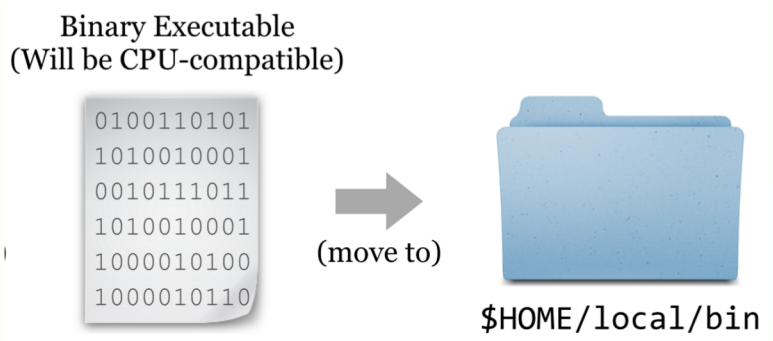


- **make** is a program that comes with Unix/Linux
- It looks for a file called **Makefile** (that came with the download or was created by **./configure**)
- **make** uses the instructions in the **Makefile** that specify how the **source** code should be turned into (compiled) into binary file(s) (1's and 0's).

Oregon State UNIVERSITY

make install

make install



- **make install** again reads the **Makefile** but instead of compiling we tell it to run the "install"
- This copies the (complied) binaries and any other support files it needs its install location.

Q: What is the **default** install location?

A: /bin, /usr/bin, /usr/local/bin

etc. These directories are owned by **root**!

Oregon State UNIVERSITY

Quiz

If you wanted to change where the default install directory was, which step would you do that in?

```
./configure
make
make check
make install
```

Oregon State UNIVERSITY

./configure -prefix

A: ./**configure**

We can see what options are available in configure

./**configure -h**

We want to use **-prefix** to specify `$HOME/local`

./**configure -prefix**=`$HOME/local`

```
make
make install
```

Note: 'bin' is implied in this case.

Oregon State UNIVERSITY

Verify install and test

```
cd $HOME/local
ls
cd bin
ls
```

Since $HOME/local/bin is in our **$path** we can run the programs from anywhere!

```
cd $HOME
where hmmsearch
hmmsearch
```

Oregon State UNIVERSITY

Command / Concept Review

- `set path`
- `wget`
- `gzip`
- `tar`
- `file`
- `./configure -prefix`
- `make (|check|install)`

`.login`

`.cshrc`

Source

Binary

Makefile

Oregon State UNIVERSITY