Shells, Environment, Scripting, and Bash

(in 80 minutes[!])

Pat Pannuto / Marcus Darden

Q: How does a program start?

Q: How does a program start?

The jobs of a shell

- Spawn (launch) new programs
- Handle input and output to programs
- Kill and clean up old programs

Q: How does a program start?

The jobs of a shell

- Spawn (launch) new programs
- Handle input and output to programs
- Kill and clean up old programs

What shells have you used?

Let's poke around how the [Desktop] shell works

```
$ cp /usr/share/applications/firefox.desktop ~/Desktop/
$ chmod +x ~/Desktop/firefox.desktop
```

Let's poke around how the [Desktop] shell works

```
$ cp /usr/share/applications/firefox.desktop ~/Desktop/
$ chmod +x ~/Desktop/firefox.desktop
```

What makes `firefox.desktop` work?

Let's poke around how the [Desktop] shell works

```
$ cp /usr/share/applications/firefox.desktop ~/Desktop/
$ chmod +x ~/Desktop/firefox.desktop
```

What makes `firefox.desktop` work?

How does the [desktop] shell:

- Spawn (launch) new programs
- Handle input and output to programs
- Kill and clean up old programs

Let's poke around how the [bash] shell works

```
$ firefox
<Ctrl-C>
$ firefox &
$ jobs
$ fg
<Ctrl-Z>
$ bg

$ echo "hello" > test
$ cat test

$ true && echo "hello"
$ false && echo "nope" || echo "whaaaat?"
```

Let's poke around how the [bash] shell works

```
$ firefox
<Ctrl-C>
$ firefox &
$ jobs
$ fg
<Ctrl-Z>
$ bg

$ echo "hello" > test
$ cat test

$ true && echo "hello"
$ false && echo "nope" || echo "whaaaat?"
```

How does the [bash] shell:

- Spawn (launch) new programs
- Handle input and output to programs
- Kill and clean up old programs

Where's `firefox` anyway?

```
$ firefox                # This works
$ gcc hello.c -o hello # This works
$ hello                  # This doesn't
$ ./hello                # This works
```

Your environment affects program behavior

- Even shells! (they're a program too)

Changing the environment will change program behavior

- In this case, how a shell performs the search for programs

```
$ PATH=$PATH:/home/username/     # Assuming "hello" is in this folder
$ hello                          # Now this works!
$ PATH=/home/username            # What if you'd done this instead?
```

- Also saw a brief example of environment variables in last week's homework

Your programs can use the environment too

```cpp
#include <iostream>
using namespace std;
int main(int argc, char *argv[], char *envp[]) {
  cout << "argc: " << argc << endl;
  cout << "envp[0]: " << envp[0] << endl;
  // while (*envp++) {  // Try this one, too!
  //     cout << *envp << endl;
  // }
  return 0;
}
```

```
$ ./a.out
$ HELLO=world ./a.out
$ lower=fine many=okaytoo ./a.out
$ export IamPermanent=ish
$ ./a.out
$ # Try uncommenting the while loop, did you find the missing ones?
$ ./a.out | less      # This may explain some of the funny colors
```

Now what about scripting?

Now what about scripting?

Surprise! You've been scripting this whole time!

- Typing commands into the bash shell and running a bash script are *the same*

```
$ cat test.sh
echo "hello" > test
cat test
true && echo "hello"
false && echo "nope" || echo "whaaaat?"
$ chmod +x test.sh # What is this doing?
$ ./test.sh
```

Now what about scripting?

Surprise! You've been scripting this whole time!

- Typing commands into the bash shell and running a bash script are *the same*

```
$ cat test.sh
echo "hello" > test
cat test
true && echo "hello"
false && echo "nope" || echo "whaaaat?"
$ chmod +x test.sh # What is this doing?
$ ./test.sh
```

- How to write a bash script?
    - Try things out in the terminal
    - Copy things that work into a file ($ history)
    - Run that file
    - Repeat

Bash is old...

But useful, especially for really short things

But has ugly and finicky syntax

- `VARIABLE=test` != `VARIABLE = test` :(

But running programs is really easy

- (it's what it was built for after all)
- `g++ -O3 -m32 thread.o libinterrupt.a test1.cpp -ldl -o test1`
- `./test1`

But doing much more is tricky

- Validate program output (`diff`?), what if it varies?
- Rule of thumb: More than 50-100 lines, more than a shell script

Closing remarks

- **Try one of the Advanced Exercises**
- We're at the end of Segment 1
- Reminder: You must submit to staff at OH