Command-Line Basics

Working at the CLI

Stepping backward?

- No! Getting rid of training wheels!
- Once comfortable:
  - Ease of working with folders and files
  - Utilize powerful Unix commands
  - Python made for this, you should be too!

Launch a Terminal

- Terminal programs vary by OS.
  - Mac: Terminal.app
  - Win: Windows Subsystem for Linux (WSL)
  - Linux: Console
- These run the program /bin/bash.
  - GNU Bourne-Again Shell
  - Reads what you type, parses commands, executes them, shows output
  - Many programs already at your disposal!

Typical Shell Workflow

1. Type a command and `<enter>`:

```
> hello this is a command
```

2. Shell parses
   - The first token is a command (“hello”).
   - The rest are arguments to that program.
3. Command is located and executed with the arguments passed in as text

Typical Shell Workflow

4. “hello” command runs
5. Any console output from “hello” is shown
6. Command completes
7. Shell waits for next command

First Commands

Here are a few good ones to start with:

`pwd [opt]`	Print the current “working directory”
`cd [opt] [dir]`	Change working directory
`ls [opt] [items]`	List details about a folder or file
`clear`	Clear the current terminal
`top [opt]`	Show running processes (‘q’ to quit)
`man [opt] <cmd>`	Show technical details for a program

[] → optional arguments

<> → required arguments

Where Are the Commands?

- A command is any “executable” file.
  - File attributes indicate if a file is executable
- On CLI: The first token is the command.
  - If there are “/” present, it is assumed to be a path to the command
- Otherwise, a PATH environment variable is used.
  - “:” separated list of folders to search

Finding Existing Commands

```
ls /bin /usr/bin /usr/local/bin
```

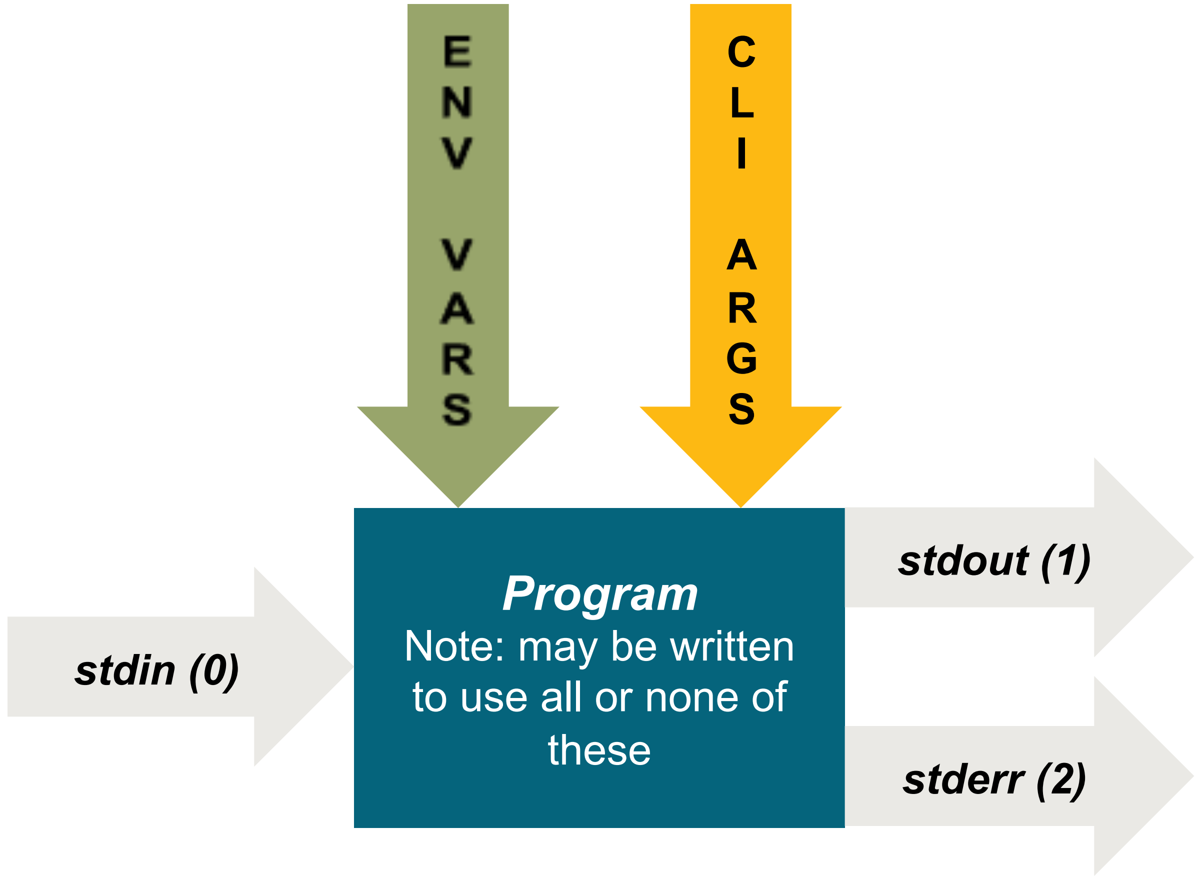
- At the CLI:
  - `man -k <keyword>`
  - Type, e.g.: `a<tab>`—this will show all commands that start with “a”
- **!!Danger!!** Some commands can delete things and/or damage your system.

What Options Exist?

- Most Unix commands have command-line options and arguments.
- How do you figure out what options are available?
  - `man <cmd>` (i.e., RT#M)
- Also, `--help` or `-h` works.

I/O Redirection and Pipelining

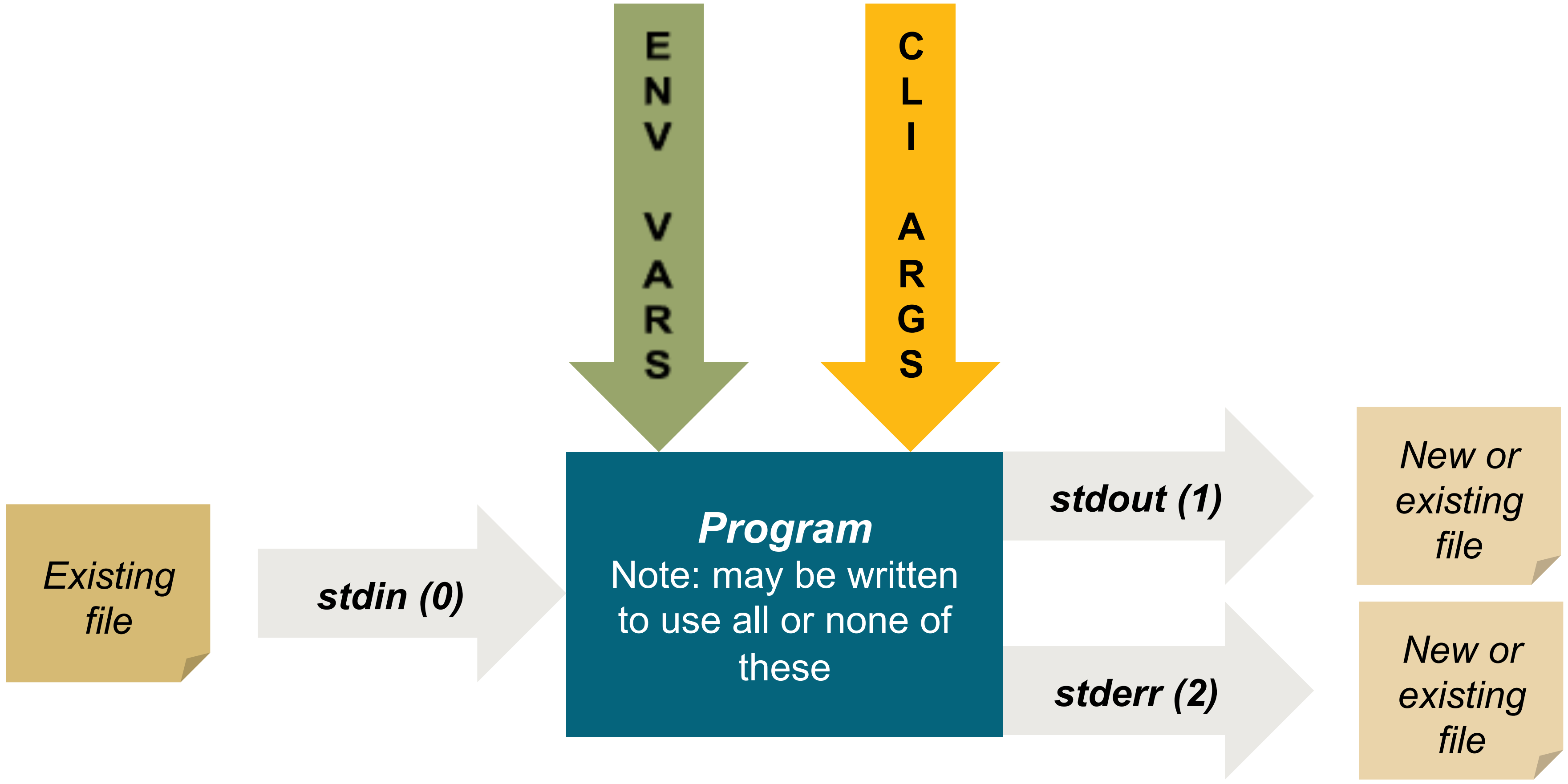
Command Input/Output



Default I/O Connections

- By default, the shell:
  - Connects the keyboard to **stdin**
  - Connects the console to **stdout**
  - Connects the console to **stderr**
- Programming choice to use these or not

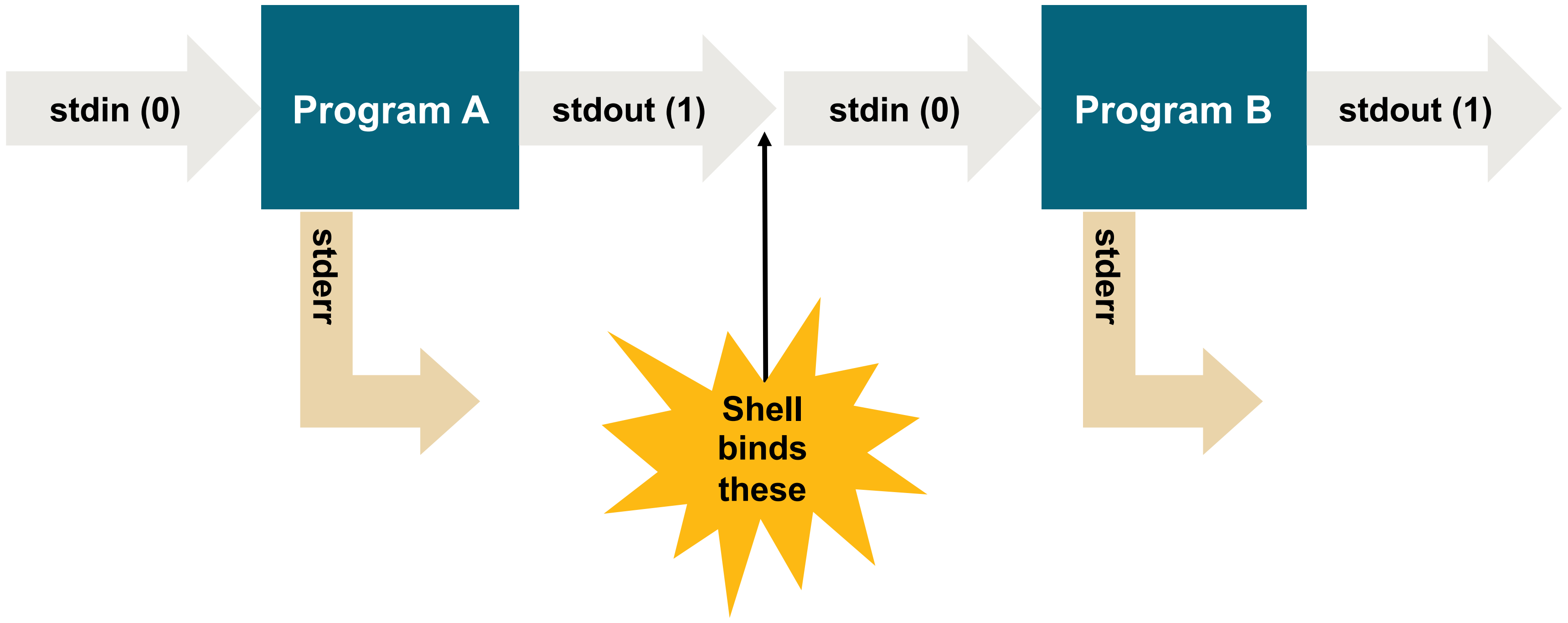
Redirecting I/O To/From Files



Redirecting I/O To/From Files

- `cmd < file` - substitute file for stdin
- `cmd > file` - substitute file for stdout (overwrite file)
- `cmd >> file` - same as `>` but append
- `cmd 2> file` - substitute file for stderr (overwrite file)
- `cmd 2>> file` - same as `2>` but append
- `cmd < file1 > file2 2> file3`
- Mix and match!

Pipelining


stdin (0)
Program A
stdout (1)
stdin (0)
Program B
stdout (1)
stderr
stderr
Shell binds these

Pipes/Pipelining

- UNIX philosophy is to have small utility programs and connect them in pipelines
- `cmd1 | cmd2`
    - (A pipe) connects the standard output of the first program to the standard input of the second.
    - `find /usr/share | less`
- Powerful!

Python
Command-Line Arguments

Running Python Programs

- `python3 [opt] hello.py [args]`
- From shell's perspective:
  - `python3` is the command
  - Everything else are arguments to `python3`
- However:
  - `[opt]` handled by the `python3`
    - `python3 –help` to see what is available
  - `hello.py` Python program to execute
  - `[args]` handled by `hello.py`

How Do We Get the Arguments?

```
## 'sys' gives us access to various system details
import sys

## sys.argv list containing the arguments

print(len(sys.argv)) ## always at least 1
print(sys.argv[0])   ## a string: program file name
print(sys.argv[1])   ## a string: IF passed in
```

- `sys` module: system-specific information
- `sys.argv` a list with at least one element
- We can choose to access these or not

Example: `Hello.Py`

```
> cat hello.py
import sys
print(sys.argv[0], ": Hello,", sys.argv[1])

> python3 hello.py
Traceback (most recent call last):
   File "hello.py", line 2, in <module>
      print(sys.argv[0], ": Hello,",
      sys.argv[1])
IndexError: list index out of range

> python3 hello.py Denver
hello.py : Hello, Denver
```