

# [220] The Terminal

Meena Syamkumar  
Mike Doescher

```
settern — -bash — 80x24
Last login: Wed Feb 24 10:56:19 on ttys003
new-host-6:~ settern$
```

```
E:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

E:\Users\ACK>shutdown /?
Usage: shutdown /l /s /r /g /a /p /h /e
        /m \\computername [/t xx] [/d [p|u]:xx:yy [/c "comment"]]

No args      Display help. This is the same as typing /?.
/?           Display help. This is the same as not typing
/i           Display the graphical user interface (GUI).
             This must be the first option.
/l           Log off. This cannot be used with /m or /d o
Shutdown the computer.
/r           Shutdown and restart the computer.
/g           Shutdown and restart the computer. After the
             rebooted, restart any registered application
/a           Abort a system shutdown.
             This can only be used during the time-out pe
Turn off the local computer with no time-out
Can be used with /d and /f options.
/h           Hibernate the local computer.
             Can be used with the /f option.
/e           Document the reason for an unexpected shutdo
/m \\computername Specify the target computer.
/t xxx       Set the time-out period before shutdown to x
             The valid range is 0-315360000 (10 years), w
             If the timeout period is greater than 0, the
             implied.
/c "comment" Comment on the reason for the restart or s
             Maximum of 512 characters allowed.
/f           Force running applications to close without
             The /f parameter is implied when a value greater than 0 is
             specified for the /t parameter.
/d [p|u]:xx:yy Provide the reason for the restart or shutdown.
             p indicates that the restart or shutdown is planned.
             u indicates that the reason is user defined.
             If neither p nor u is specified the restart or shutdown is
             unplanned.
             xx is the major reason number (positive integer less than 256).
             yy is the minor reason number (positive integer less than 65536).
```

```
Windows PowerShell
Copyright (C) 2009 Microsoft Corporation. All rights reserved.

PS C:\Users\NullByte> get-help
Get-Help

SHORT DESCRIPTION
    Displays help about Windows PowerShell cmdlets and concepts.

LONG DESCRIPTION
    ...

SYNTAX
    get-help <CmdletName> [-TopicName]
    help <CmdletName> [-TopicName]
    <CmdletName> -?

    "Get-help" and "-?" display help on one page.
    "Help" displays help on multiple pages.

Examples:
    get-help get-process      : Displays help about the get-process cmdlet.
    get-help about_signing    : Displays help about digital signatures.
    help where-object         : Displays help about the where-object cmdlet.
    help about_foreach        : Displays help about the foreach-object cmdlet.
    set-service -?            : Displays help about the set-service cmdlet.

You can use wildcard characters in the help topics. If only one help topic matches, PowerShell displays the help for that topic.

Examples:
    get-help *                : Displays all help topics.
    get-help get-*            : Displays topics that begin with "get-".
    help where-object*        : Displays topics with "where-object" in the name.
    get-help about*           : Displays all conceptual help topics.

For information about wildcards, type:
    get-help about_wildcard

REMARKS
    To learn about Windows PowerShell, read the
    get-command : Gets information about cmdlets.
    get-member  : Gets the properties and methods of an object.
    where-object : Filters object properties.
    about_object : Explains the use of object.
    about_remote : Tells how to run commands.
```

```
hello world
stuart@stuart-desktop:~$
```

# Today's Topics

## Terminal Emulators and Shells

- Terminal history
- Shells
- Running programs from a shell

## Navigation

## Running Programs and Commands

## Demos

# History: the original terminals



**Mainframe  
(powerful computer)**



**How to share it?**



**dumb terminals  
(text based)**



# Terminal emulators

```
ty-mac:var$ ls -la
total 0
drwxr-xr-x 26 root    wheel    832 Dec 26 2017 .
drwxr-xr-x  6 root    wheel   192 Dec 26 2017 ..
drwx----- 2 root    wheel    64 Jul 15 2017 agentx
drwxr-xr-x  8 daemon  wheel   256 Dec  1 2017 at
drwx----- 77 root    wheel  2464 Jul 30 09:48 audit
drwx----- 2 root    wheel    64 Oct  2 2017 backups
drwxr-xr-x 91 root    wheel  2912 Aug 15 23:40 db
drwxr-xr-x  2 root    sys      64 Oct  2 2017 empty
drwxr-xr-x  4 root    wheel   128 Apr 27 2016 folders
drwx----- 2 root    wheel    64 Jul 28 23:19 install
drwxr-x---  2 _jabber  _jabber  64 Jul 15 2017 jabberd
drwxr-xr-x  3 root    wheel    96 Jul 25 2017 lib
drwxr-xr-x 46 root    wheel  1472 Aug 17 14:41 log
drwxr-x---  2 _mobileasset _mobileasset 64 Oct  2 2017 ma
drwxr-xr-x  3 root    wheel    96 Oct  2 2017 msgs
drwxr-xr-x  2 root    wheel    64 Oct  2 2017 netboot
drwxr-xr-x  6 _networkd _networkd 192 Dec 26 2017 networkd
drwxr-x---  7 root    wheel   224 May  9 2016 root
drwxr-xr-x  4 root    wheel   128 Jul 15 2017 rpc
drwxrwxr-x 30 root    daemon  960 Aug 16 15:07 run
drwxr-xr-x  2 daemon  wheel    64 Oct  2 2017 rwho
drwxr-xr-x  6 root    wheel   192 Dec  1 2017 spool
drwxrwxrwt  5 root    wheel   160 Jul 30 10:40 tmp
drwxr-xr-x  4 root    wheel   128 Aug 16 09:51 vm
drwxr-xr-x  4 root    wheel   128 Dec 26 2017 yp
```



why???



fast



slow



local computer  
(e.g., personal)

# Terminal emulators

Career Tip 1: know the difference between **familiar** tools and **good** tools

Practice using good tools that are unfamiliar

Investment is more important than working hard

```
drwxr-xr-x  2 daemon  wheel   64 Oct  2  2017 rwho
drwxr-xr-x  6 root    wheel  192 Dec  1  2017 spool
drwxrwxrwt  5 root    wheel  160 Jul 30 10:40 tmp
drwxr-xr-x  4 root    wheel  128 Aug 16 09:51 vm
drwxr-xr-x  4 root    wheel  128 Dec 26 2017 yp
```



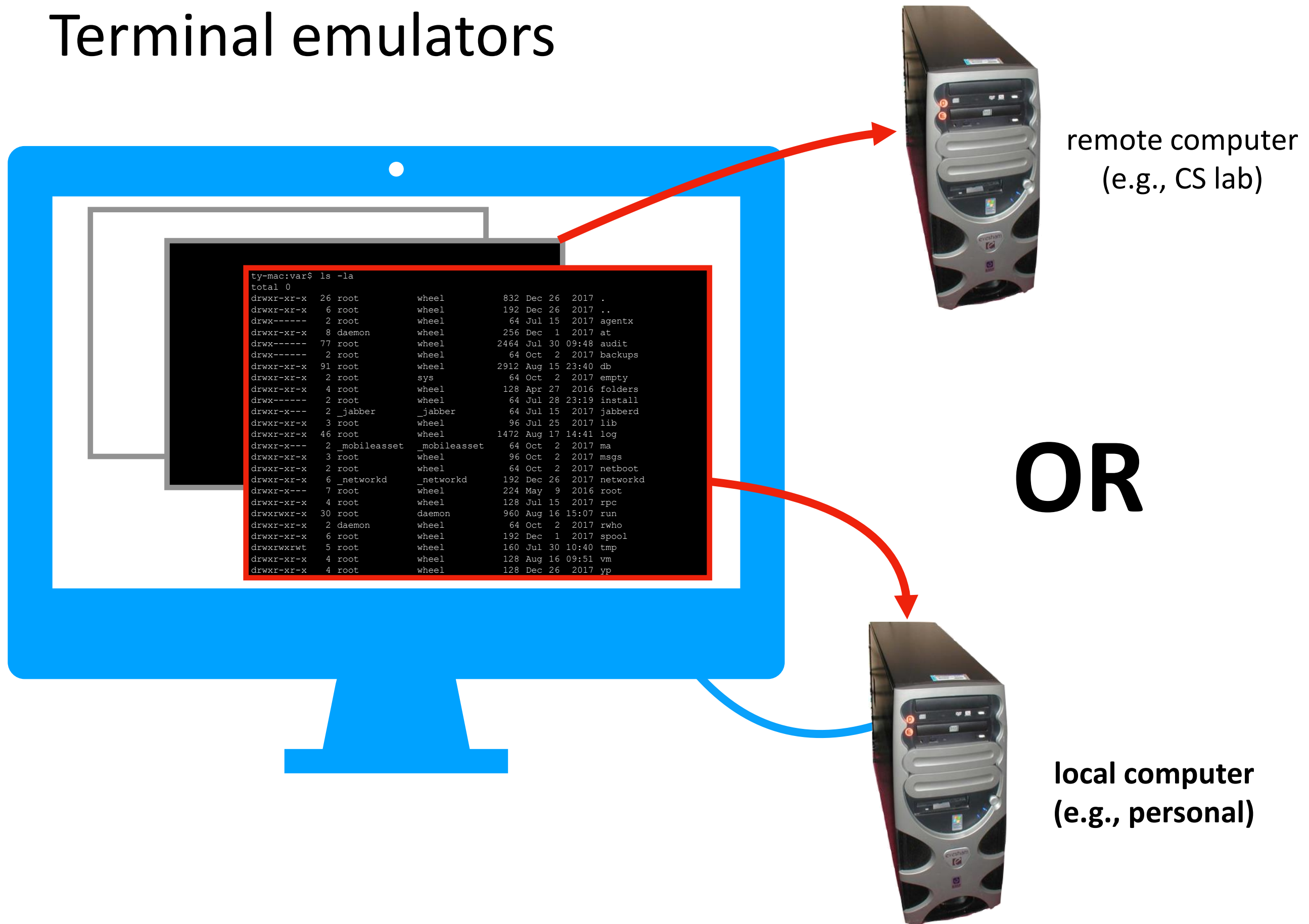
slow



local computer  
(e.g., personal)



# Terminal emulators



# Terminal emulators

Career Tip 2: master the tools that let you work from anywhere

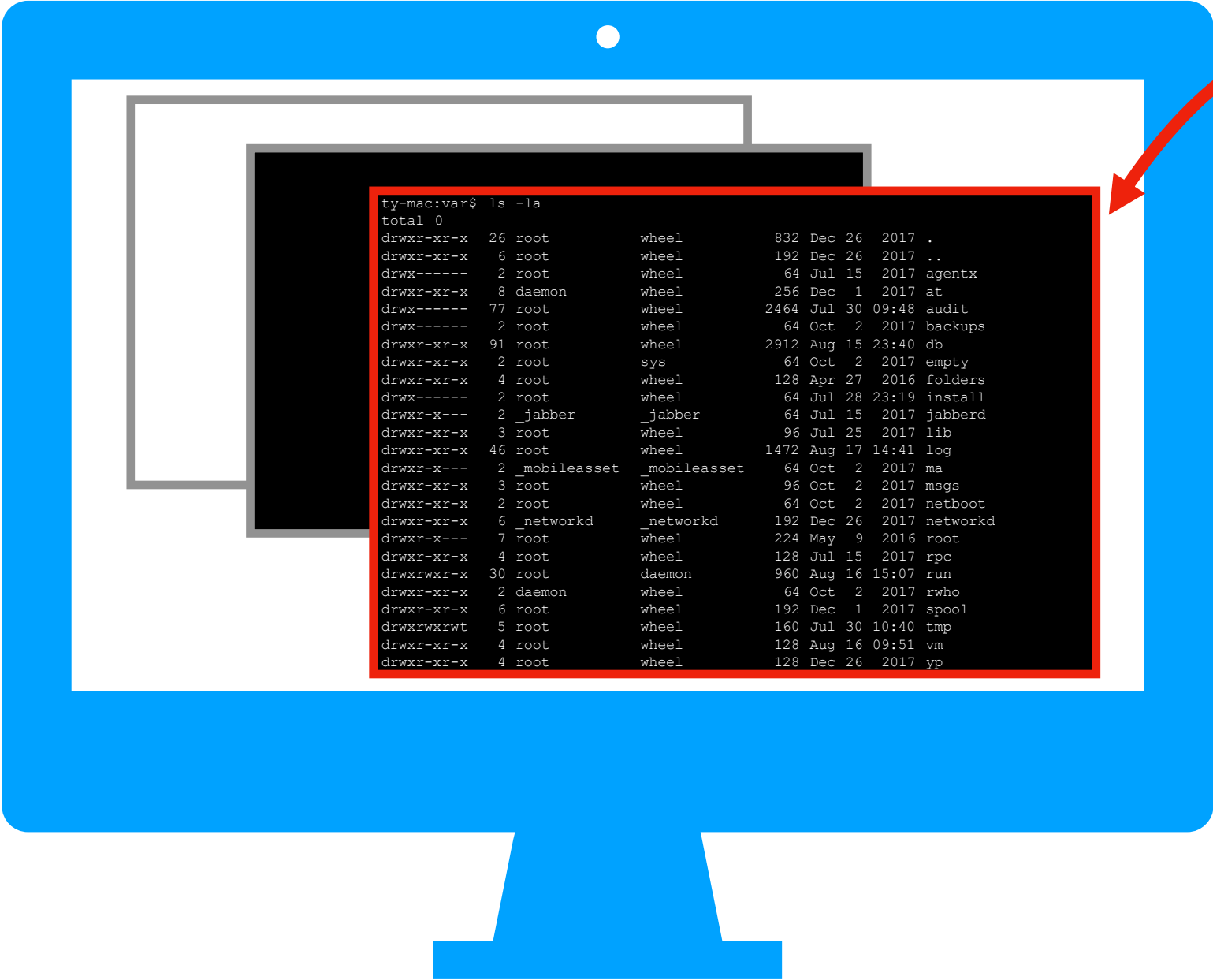
Work for the highest-paying place from the most enjoyable place (home? beach?)



<https://www.cnn.com/travel/article/australia-best-beaches/index.html>

# Terminal emulators

programming running in  
the terminal emulator  
is called a "shell"



```
ty-mac:var$ ls -la
total 0
drwxr-xr-x  26 root      wheel      832 Dec 26  2017 .
drwxr-xr-x   6 root      wheel     192 Dec 26  2017 ..
drwx-----  2 root      wheel       64 Jul 15  2017 agentx
drwxr-xr-x   8 daemon    wheel     256 Dec  1  2017 at
drwx----- 77 root      wheel    2464 Jul 30  09:48 audit
drwx-----  2 root      wheel       64 Oct  2  2017 backups
drwxr-xr-x  91 root      wheel    2912 Aug 15  23:40 db
drwxr-xr-x   2 root      sys        64 Oct  2  2017 empty
drwxr-xr-x   4 root      wheel     128 Apr 27  2016 folders
drwx-----  2 root      wheel       64 Jul 28  23:19 install
drwxr-x---   2 _jabber    _jabber    64 Jul 15  2017 jabberd
drwxr-xr-x   3 root      wheel       96 Jul 25  2017 lib
drwxr-xr-x  46 root      wheel    1472 Aug 17  14:41 log
drwxr-x---   2 _mobileasset _mobileasset 64 Oct  2  2017 ma
drwxr-xr-x   3 root      wheel       96 Oct  2  2017 msgs
drwxr-xr-x   2 root      wheel       64 Oct  2  2017 netboot
drwxr-xr-x   6 _networkd  _networkd 192 Dec 26  2017 networkd
drwxr-x---   7 root      wheel     224 May  9  2016 root
drwxr-xr-x   4 root      wheel     128 Jul 15  2017 rpc
drwxrwxr-x  30 root      daemon    960 Aug 16  15:07 run
drwxr-xr-x   2 daemon    wheel       64 Oct  2  2017 rwho
drwxr-xr-x   6 root      wheel     192 Dec  1  2017 spool
drwxrwxrwt   5 root      wheel     160 Jul 30  10:40 tmp
drwxr-xr-x   4 root      wheel     128 Aug 16  09:51 vm
drwxr-xr-x   4 root      wheel     128 Dec 26  2017 yp
```



# Today's Topics

## Terminal Emulators and Shells

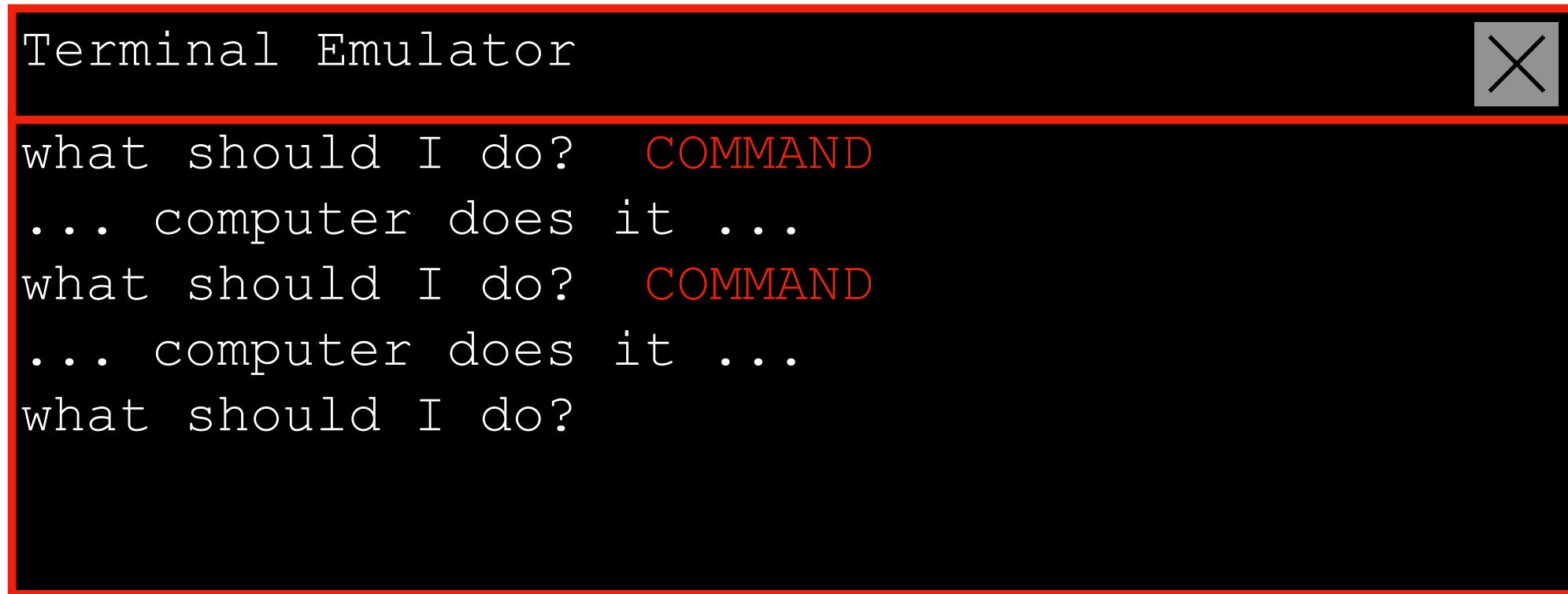
- Terminal history
- Shells
- Running programs from a shell

## Navigation

## Running Programs and Commands

## Demos

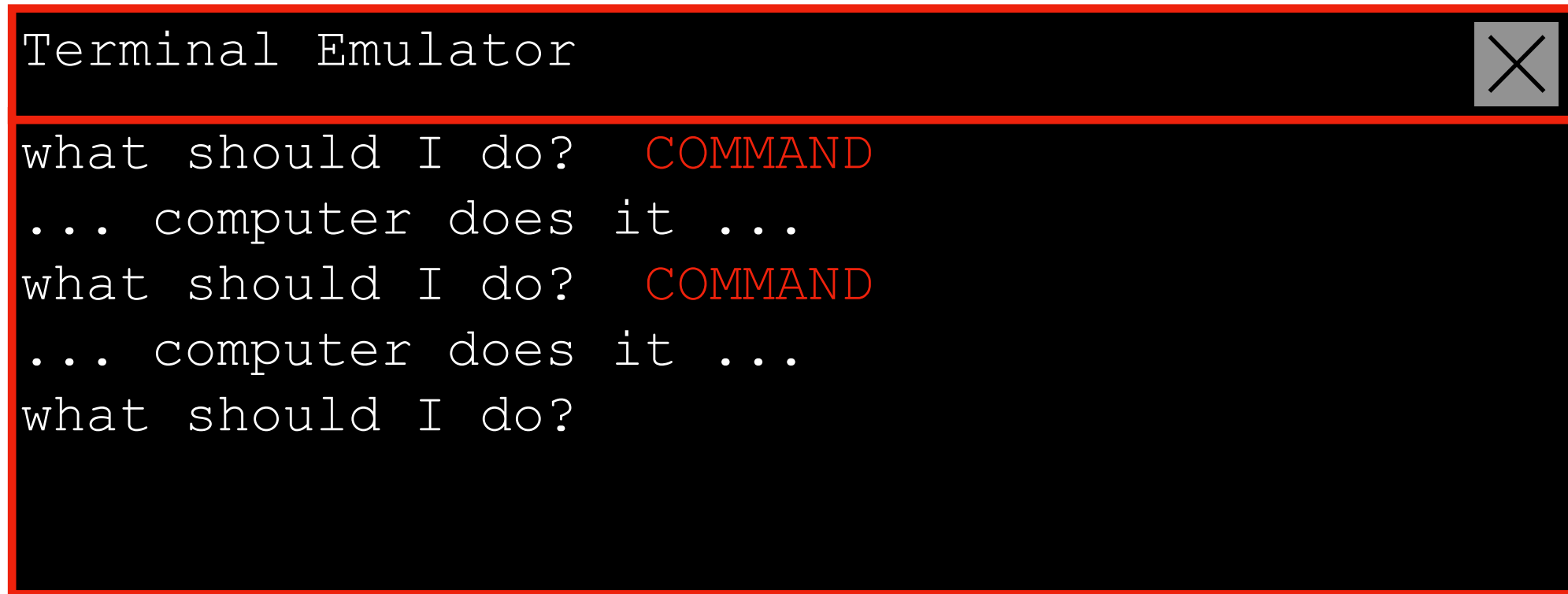
# Shell: the most helpful program

A screenshot of a terminal emulator window with a black background and a red border. The title bar at the top says "Terminal Emulator" and has a close button on the right. The text inside the terminal is white, with the word "COMMAND" appearing in red. The text shows a sequence of prompts and responses: "what should I do?", "... computer does it ...", "what should I do?", "... computer does it ...", and "what should I do?".

```
Terminal Emulator
what should I do?  COMMAND
... computer does it ...
what should I do?  COMMAND
... computer does it ...
what should I do?
```

- 1 **navigate:** dig through folders and files
- 2 **run programs**

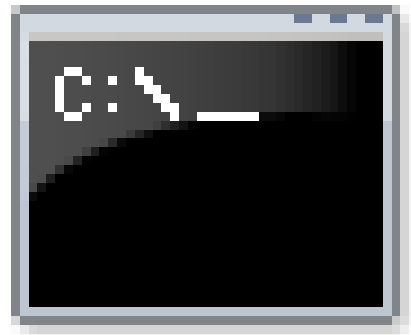
# Shell: the most helpful program

A screenshot of a terminal emulator window with a black background and a red border. The title bar at the top says "Terminal Emulator" and has a close button on the right. The text inside the terminal is white, with the word "COMMAND" appearing in red. The text reads: "what should I do? COMMAND", "... computer does it ...", "what should I do? COMMAND", "... computer does it ...", and "what should I do?".

```
Terminal Emulator  
what should I do?  COMMAND  
... computer does it ...  
what should I do?  COMMAND  
... computer does it ...  
what should I do?
```

- 1 **navigate:** dig through folders directories and files
- 2 **run programs**

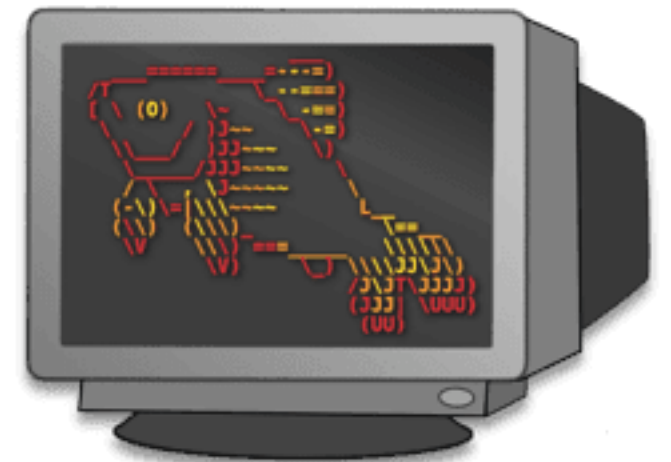
You have a few options when it comes to shells...



cmd



PowerShell



fish



ksh

today

zsh

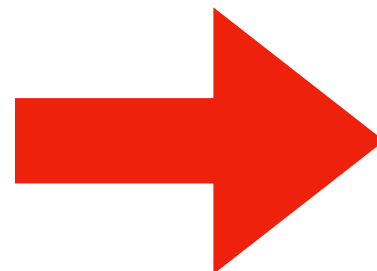
csh



Stephen Bourne

**/bin/sh**  
**Bourne Shell**

1979



**BASH**  
THE BOURNE-AGAIN SHELL

# Today's Topics

## Terminal Emulators and Shells

- Terminal history
- Shells
- Running programs from a shell

## Navigation

## Running Programs and Commands

## Demos



# Running Programs

Running programs is easy, just type name of the program and hit enter:

```
ty-mac:var$
```

# Running Programs

Running programs is easy, just type name of the program and hit enter:

```
ty-mac:var$ ls
```

# Running Programs

Running programs is easy, just type name of the program and hit enter:

```
ty-mac:var$ ls
agentx      jabberd     root
at          lib         rpc
audit       log         run
backups     ma          rwho

ty-mac:var$
```

# Running Programs

Running programs is easy, just type name of the program and hit enter:

program name

```
prompt ty-mac:var$ ls
```

```
output agentx      jabberd      root
      at         lib         rpc
      audit      log         run
      backups    ma         rwho
```

```
prompt ty-mac:var$
```

a "prompt" is the question, *what should I do?*

# Today's Topics

## Terminal Emulators and Shells

### Navigation

- Storage Drives (Windows)
- Files
- Directories (aka Folders)
- Windows vs. Mac

## Running Programs and Commands

## Demos

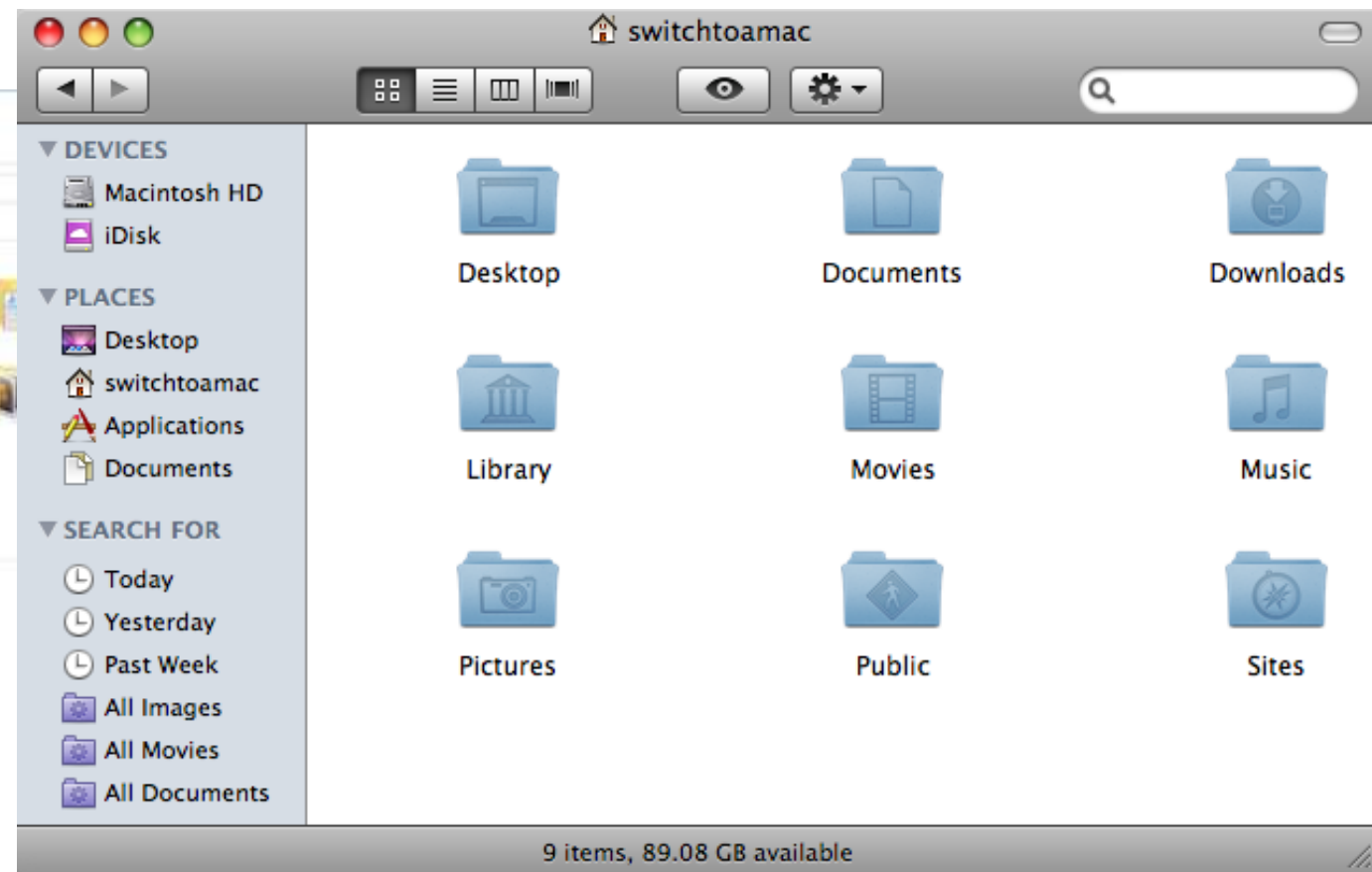
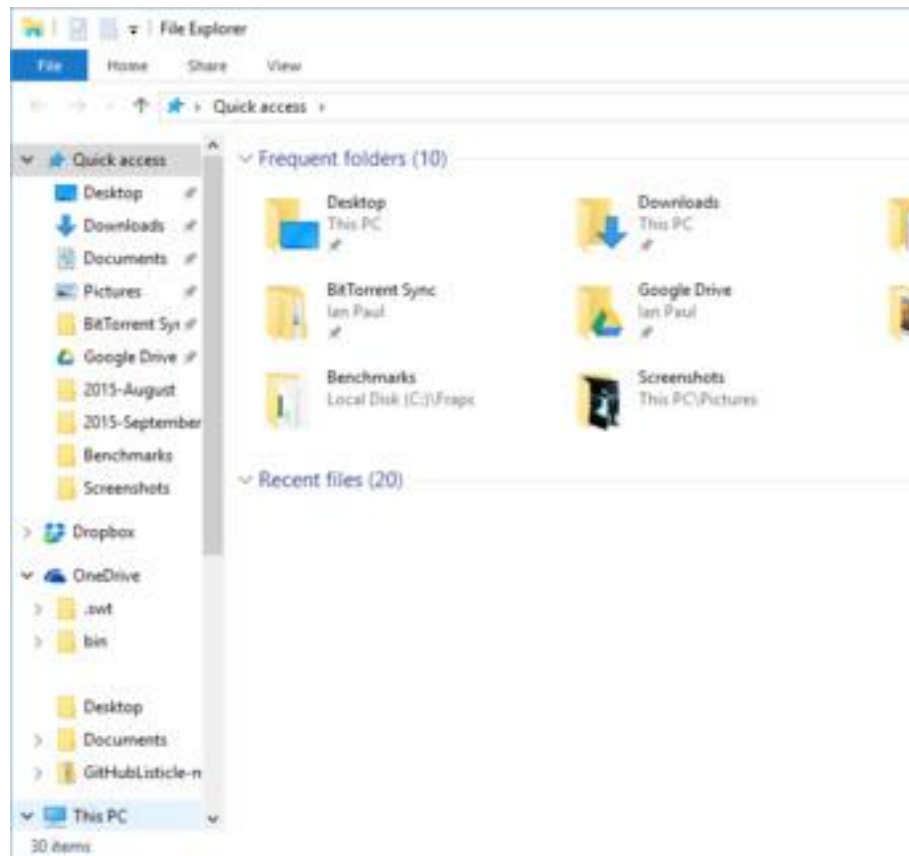


# What is navigation?

Navigation is looking around for files/folders you want

## Navigation programs

- File Explorer (Windows)
- Finder (Mac)



# What is navigation?

Navigation is looking around for files/folders you want

Navigation programs

- File Explorer (Windows)
- Finder (Mac)

With shell, navigate w/ various commands...

ls

pwd

cat

...

cd

mkdir

# Today's Topics

## Terminal Emulators and Shells

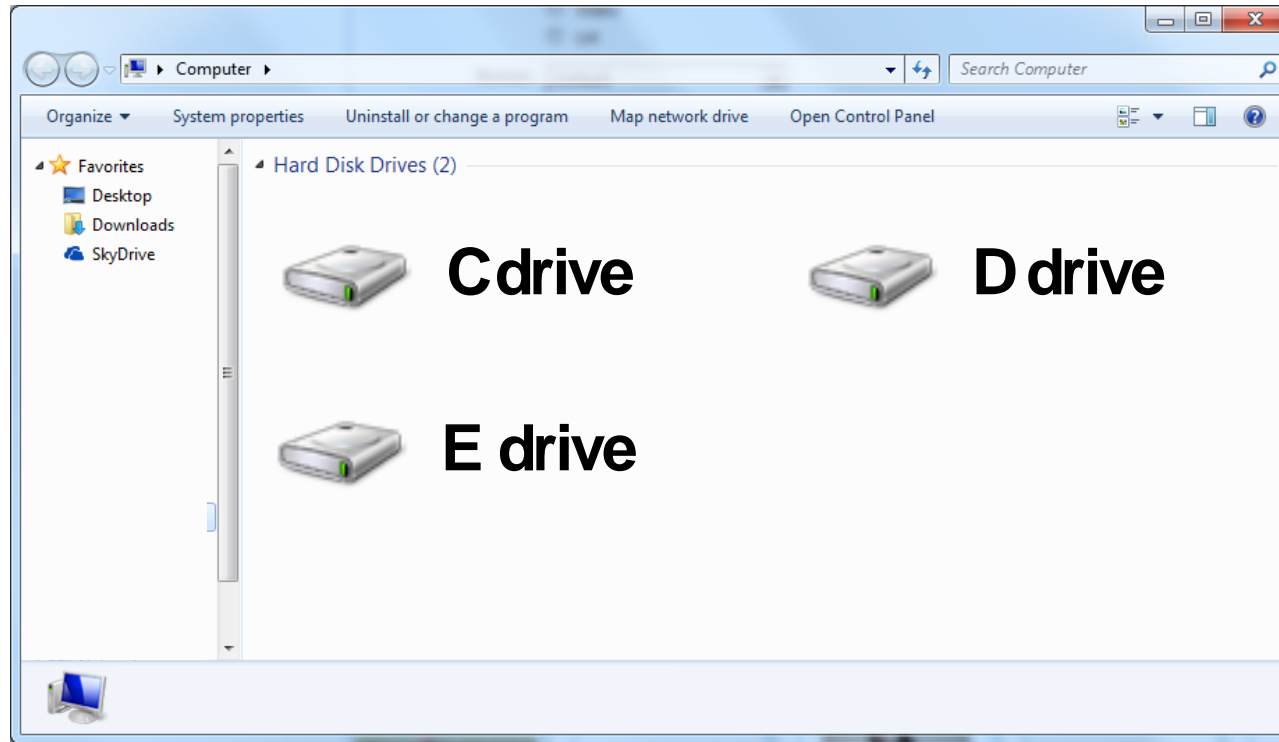
### Navigation

- Storage Drives (Windows)
- Files
- Directories (aka Folders)
- Windows vs. Mac

## Running Programs and Commands

### Demos

# Windows Storage Drives



**Each added drive is given  
its own drive letter**



# Today's Topics

## Terminal Emulators and Shells

### Navigation

- Storage Drives (Windows)
- Files
- Directories (aka Folders)
- Windows vs. Mac

## Running Programs and Commands

### Demos



# Files

Each file has a name, called a “path name”

**c:\README.txt**

**c:\hw.docx**

**d:\page.html**

**e:\main.py**

# Files

Each file has a name, called a “path name”

filename  
  
**c:\README.txt**

c:\hw.docx

d:\page.html

e:\main.py

# Files

Each file has a name, called a “path name”

filename  
c:\README.txt  
pathname

c:\hw.docx

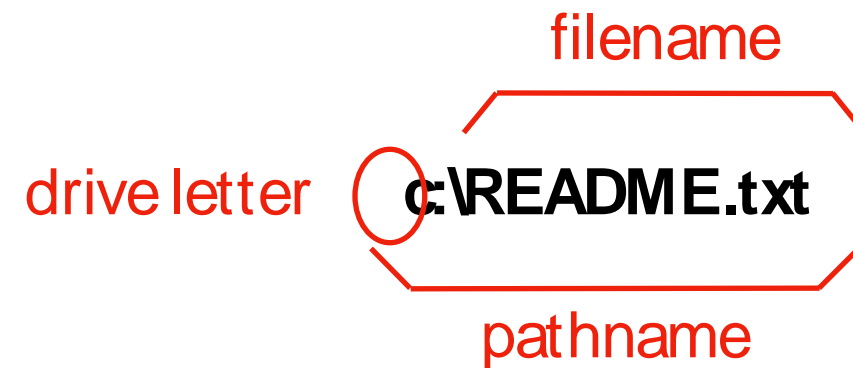
d:\page.html

e:\main.py

# Files

Each file has a name, called a “path name”

filename  
drive letter **c:\README.txt**  
pathname



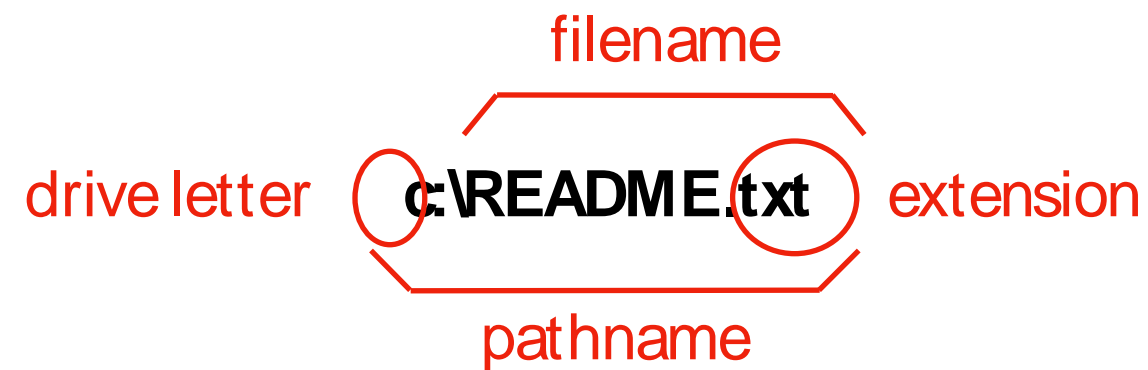
c:\hw.docx

d:\page.html

e:\main.py

# Files

Each file has a name, called a “path name”



`c:\hw.docx`

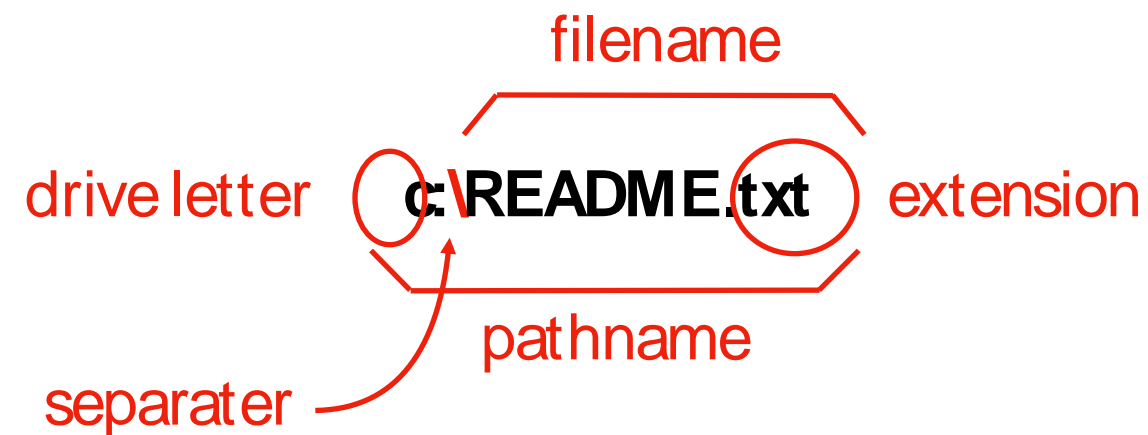
`d:\page.html`

`e:\main.py`



# Files

Each file has a name, called a “path name”



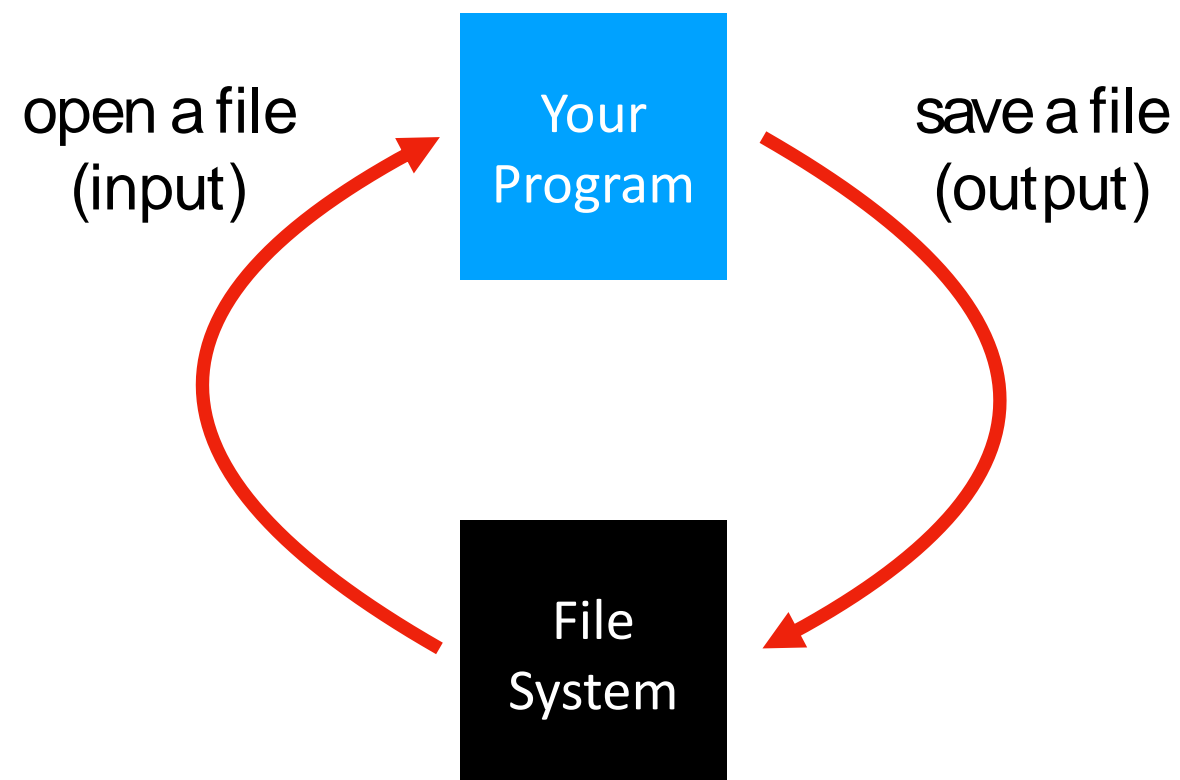
`c:\hw.docx`

`d:\page.html`

`e:\main.py`

# Files

Files might be either **input** or **output** for your programs



# Today's Topics

## Terminal Emulators and Shells

### Navigation

- Storage Drives (Windows)
- Files
- Directories (aka Folders)
- Windows vs. Mac

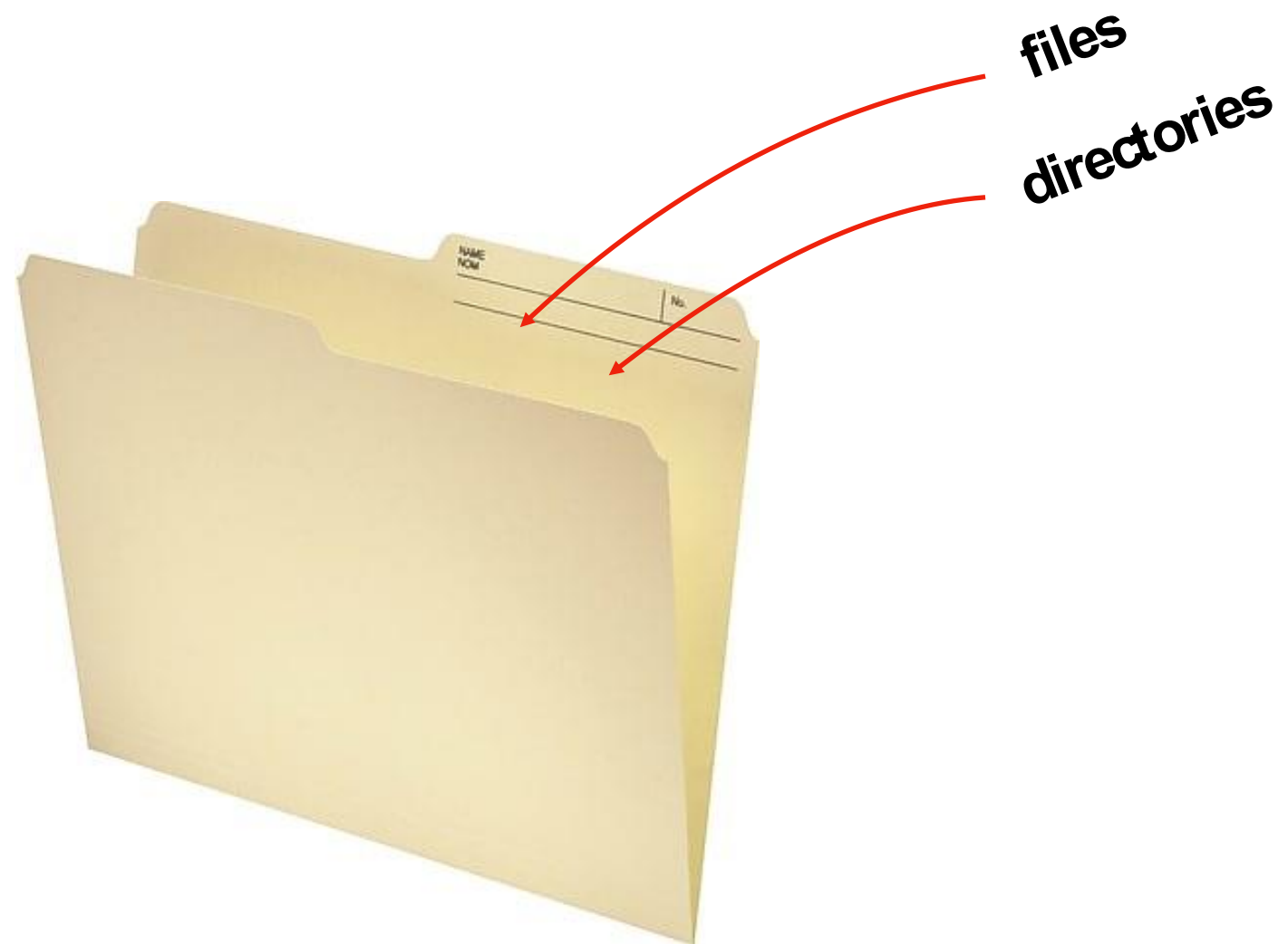
## Running Programs and Commands

### Demos

# Directories

Directories are used to organize files and sub directories

- Also called “folders”
- A directory also has pathname



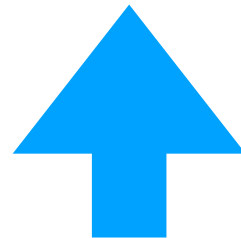
# Directories

Directories are used to organize files and sub directories

- Also called “folders”
- A directory also has pathname

Example paths:

- c:\my-directory\file1.docx
- c:\my-directory\file2.docx
- c:\my-directory\file3.docx



in my-directory

# Directories

Directories are used to organize files and sub directories

- Also called “folders”
- A directory also has pathname

Example paths:

- c:\my-directory\file1.docx
- c:\my-directory\file2.docx
- c:\my-directory\file3.docx
- c:\directory1\directory2\file1.docx
- c:\same-dir\same-dir\readme.txt

two types of paths: relative or absolute

# Relative Paths

*Where is the Computer Science building?*

- **Answer 1:** 1210 W Dayton St, Madison, WI 53706
- **Answer 2:** on the other side of Johnson street



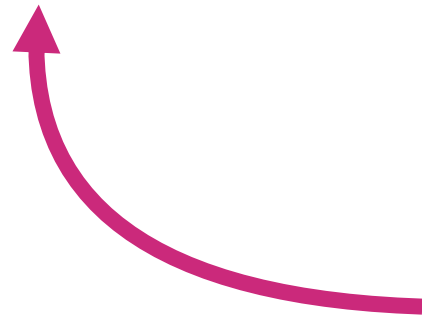
When is Answer 2 appropriate?

- When you're in the psychology building
- It may be more convenient

# Relative Paths

*Where is the Computer Science building?*

- **Answer 1:** 1210 W Dayton St, Madison, WI 53706
- **Answer 2:** on the other side of Johnson street



When is Answer 2 appropriate?

- When you're in the psychology building
- It may be more convenient

Pathnames are absolute (answer 1) or relative (answer 2)

- Absolute paths: always possible
- Relative paths: **if current location is known**
- **Working Directory** (our current location)



# Absolute vs. Relative

Absolute Path	Working Directory	Relative Path
c:\test.txt	c:\	test.txt
c:\x\y\z\my.docx	c:\x\y\z	
c:\x\y\z\my.docx	c:\x\y	
c:\x\y\z	c:\x	

# Absolute vs. Relative

Absolute Path	Working Directory	Relative Path
c:\test.txt	c:\	test.txt
c:\x\y\z\my.docx	c:\x\y\z	my.docx
c:\x\y\z\my.docx	c:\x\y	z\my.docx
c:\x\y\z	c:\x	y\z

## Two special directory names

- “..” means up a directory
- “.” means current directory

# Absolute vs. Relative

Absolute Path	Working Directory	Relative Path
c:\test.txt	c:\	test.txt
c:\x\y\z\my.docx	c:\x\y\z	my.docx
c:\x\y\z\my.docx	c:\x\y	z\my.docx
c:\x\y\z	c:\x	y\z
c:\test.txt	c:\	.\test.txt
c:\test.txt	c:\	..\test.txt
c:\x\y\z	c:\x	.\y\z
c:\x	c:\x\y\z	

## Two special directory names

- “..” means up a directory
- “.” means current directory

# Absolute vs. Relative

Absolute Path	Working Directory	Relative Path
c:\test.txt	c:\	test.txt
c:\x\y\z\my.docx	c:\x\y\z	my.docx
c:\x\y\z\my.docx	c:\x\y	z\my.docx
c:\x\y\z	c:\x	y\z
c:\test.txt	c:\	.\test.txt
c:\test.txt	c:\	..\test.txt
c:\x\y\z	c:\x	.\y\z
c:\x	c:\x\y\z	..\..

## Two special directory names

- “.” means up a directory
- “.” means current directory

# Absolute vs. Relative

Absolute Path	Working Directory	Relative Path
c:\test.txt	c:\	test.txt
c:\x\y\z\my.docx	c:\x\y\z	my.docx
c:\x\y\z\my.docx	c:\x\y	z\my.docx
c:\x\y\z	c:\x	y\z
c:\test.txt	c:\	.\test.txt
c:\test.txt	c:\	..\test.txt
c:\x\y\z	c:\x	.\y\z
c:\x	c:\x\y\z	..\..
c:\B\file.txt	c:\A	

## Two special directory names

- “.” means up a directory
- “..” means current directory

# Absolute vs. Relative

Absolute Path	Working Directory	Relative Path
c:\test.txt	c:\	test.txt
c:\x\y\z\my.docx	c:\x\y\z	my.docx
c:\x\y\z\my.docx	c:\x\y	z\my.docx
c:\x\y\z	c:\x	y\z
c:\test.txt	c:\	.\test.txt
c:\test.txt	c:\	..\test.txt
c:\x\y\z	c:\x	.\y\z
c:\x	c:\x\y\z	..\..
c:\B\file.txt	c:\A	..\B\file.txt

## Two special directory names

- “..” means up a directory
- “.” means current directory

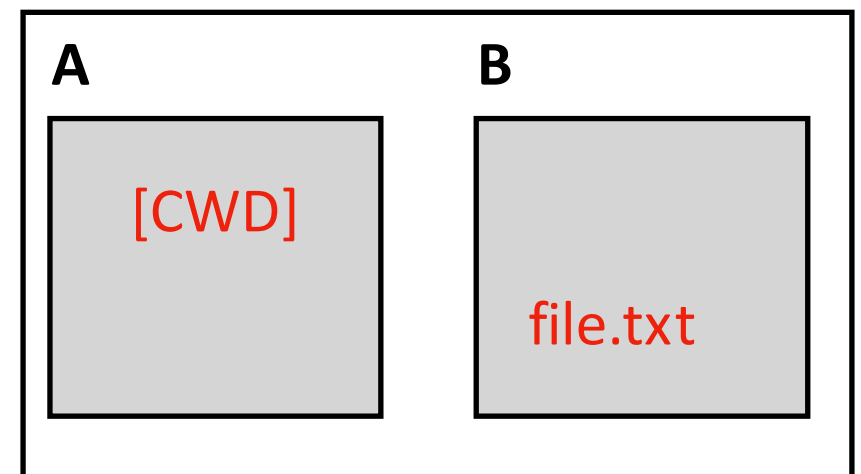
# Absolute vs. Relative

Absolute Path	Working Directory	Relative Path
c:\test.txt	c:\	test.txt
c:\x\y\z\my.docx	c:\x\y\z	my.docx
c:\x\y\z\my.docx	c:\x\y	z\my.docx
c:\x\y\z	c:\x	y\z
c:\test.txt	c:\	.\test.txt
c:\test.txt	c:\	..\test.txt
c:\x\y\z	c:\x	.\y\z
c:\x	c:\x\y\z	..\..
c:\B\file.txt	c:\A	..\B\file.txt

## Two special directory names

- “..” means up a directory
- “.” means current directory

c:\

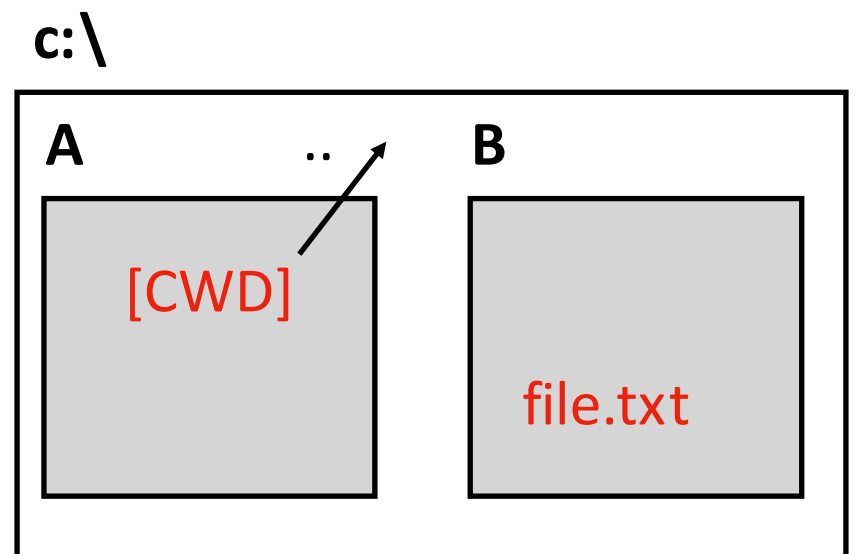


# Absolute vs. Relative

Absolute Path	Working Directory	Relative Path
c:\test.txt	c:\	test.txt
c:\x\y\z\my.docx	c:\x\y\z	my.docx
c:\x\y\z\my.docx	c:\x\y	z\my.docx
c:\x\y\z	c:\x	y\z
c:\test.txt	c:\	.\test.txt
c:\test.txt	c:\	..\test.txt
c:\x\y\z	c:\x	.\y\z
c:\x	c:\x\y\z	..\..
c:\B\file.txt	c:\A	..\B\file.txt

## Two special directory names

- “..” means up a directory
- “.” means current directory



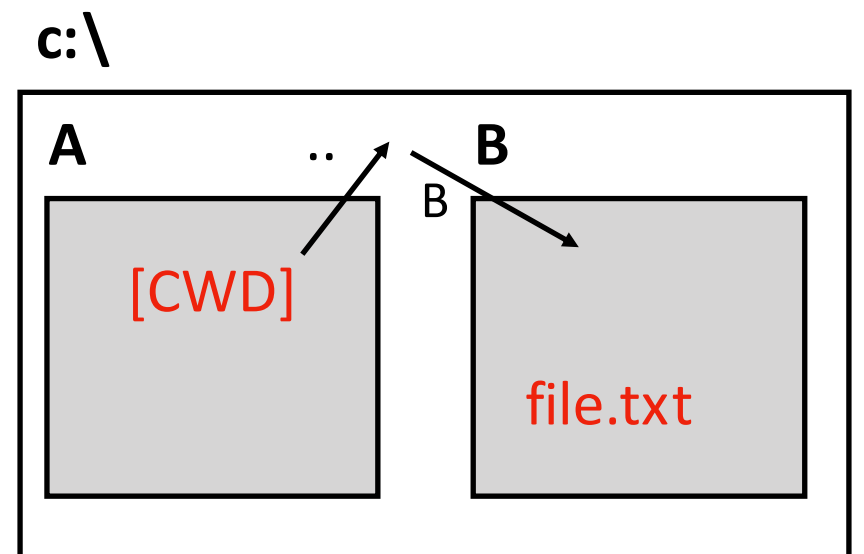


# Absolute vs. Relative

Absolute Path	Working Directory	Relative Path
c:\test.txt	c:\	test.txt
c:\x\y\z\my.docx	c:\x\y\z	my.docx
c:\x\y\z\my.docx	c:\x\y	z\my.docx
c:\x\y\z	c:\x	y\z
c:\test.txt	c:\	.\test.txt
c:\test.txt	c:\	..\test.txt
c:\x\y\z	c:\x	.\y\z
c:\x	c:\x\y\z	..\..
c:\B\file.txt	c:\A	..\B\file.txt

## Two special directory names

- “..” means up a directory
- “.” means current directory

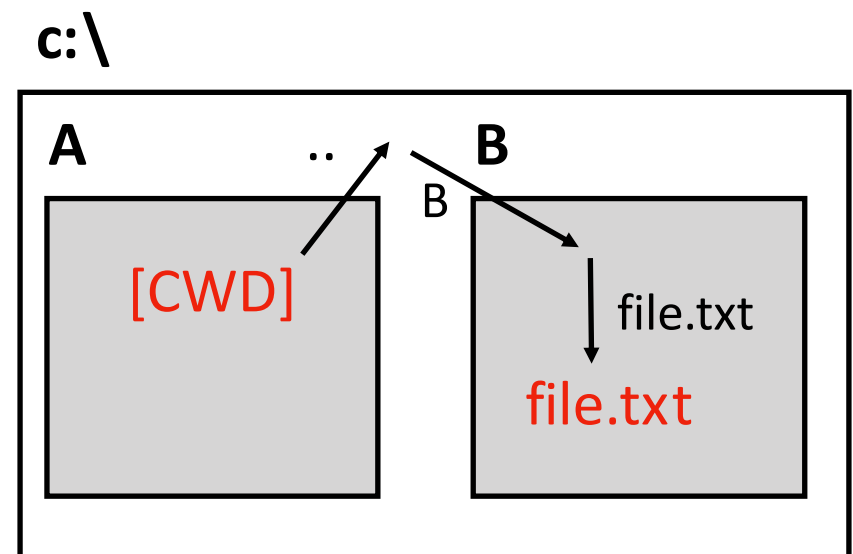


# Absolute vs. Relative

Absolute Path	Working Directory	Relative Path
c:\test.txt	c:\	test.txt
c:\x\y\z\my.docx	c:\x\y\z	my.docx
c:\x\y\z\my.docx	c:\x\y	z\my.docx
c:\x\y\z	c:\x	y\z
c:\test.txt	c:\	.\test.txt
c:\test.txt	c:\	..\test.txt
c:\x\y\z	c:\x	.\y\z
c:\x	c:\x\y\z	..\..
c:\B\file.txt	c:\A	..\B\file.txt

## Two special directory names

- “..” means up a directory
- “.” means current directory



# Absolute vs. Relative

Absolute Path	Working Directory	Relative Path
c:\test.txt	c:\	test.txt
c:\x\y\z\my.docx	c:\x\y\z	my.docx
c:\x\y\z\my.docx	c:\x\y	z\my.docx
c:\x\y\z	c:\x	y\z
c:\test.txt	c:\	.\test.txt
c:\test.txt	c:\	..\test.txt
c:\x\y\z	c:\x	.\y\z
c:\x	c:\x\y\z	..\..
c:\B\file.txt	c:\A	..\B\file.txt

## Two special directory names

- “..” means up a directory
- “.” means current directory

**more examples in demo later...**

# Today's Topics

## Terminal Emulators and Shells

### Navigation

- Storage Drives (Windows)
- Files
- Directories (aka Folders)
- Windows vs. Mac

## Running Programs and Commands

### Demos

# Multiple Drives in Mac

## Windows

- Absolute paths start with **c:\** or **d:\**
- Indicates which drive

## Mac

- Absolute paths start with **/**
- Example: **/Users/tyler/my-file.docx**
- Don't know which drive

How can we use multiple drives if every file paths starts the same???

**/.....**

**Answer: different drives feel like different directories**

# Comparison

on a Mac, a path doesn't tell you  
what drive you're on

## Windows

## Mac

## Drives

c:\Users\tyler\file.txt

/Users/tyler/file.txt

c:\Program Files

/usr/local/bin

c:\Windows\...\Logs

/var/log



d:\

/Volumes/backup

d:\A

/Volumes/backup/A



e:\movies

/Volumes/movies

e:\movies\demo1.mov

/Volumes/movies/demo1.mov



# Today's Topics

Terminal Emulators and Shells

Navigation

## Running Programs and Commands

- Navigational commands
- Arguments
- Saving output

Demos

We'll cover a few simple examples for reference in the slides, then go into more detail in the demo...

Most of these examples work in both **PowerShell** (Windows) and **bash** (Mac)



# Today's Topics

Terminal Emulators and Shells

Navigation

Running Programs and Commands

- Navigational commands
- Arguments
- Saving output

Demos

# Where am I? (What directory am I in?)

Command: `pwd`

“print working directory”

```
PS /Users/trh/scratch> pwd
```

# Where am I? (What directory am I in?)

Command: `pwd`

```
PS /Users/trh/scratch> pwd
```

```
Path
```

```
----
```

```
/Users/trh/scratch
```



this is the current directory

```
PS /Users/trh/scratch>
```

# Go up a directory

Command: `cd ..`

```
PS /Users/trh/scratch> pwd
```

```
Path
```

```
----
```

```
/Users/trh/scratch
```

```
PS /Users/trh/scratch> cd ..
```

# Go up a directory

Command: `cd ..`

```
PS /Users/trh/scratch> pwd
```

```
Path
```

```
----
```

```
/Users/trh/scratch
```

```
PS /Users/trh/scratch> cd ..
```

```
PS /Users/trh>
```

# Clear the screen

Command: `clear`

```
PS /Users/trh/scratch> pwd
```

```
Path
```

```
----
```

```
/Users/trh/scratch
```

```
PS /Users/trh/scratch> cd ..
```

```
PS /Users/trh> clear
```

# Clear the screen

Command: `clear`

```
PS /Users/trh>
```

# Go inside a directory

Command: `cd directory-name`

name of directory we started in

```
PS /Users/trh> cd scratch
```



# Go inside a directory

Command: `cd directory-name`

```
PS /Users/trh> cd scratch  
PS /Users/trh/scratch>
```

# Go to top directory

Command: `cd /`

*is this Windows or Mac?*

```
PS /Users/trh> cd scratch  
PS /Users/trh/scratch> cd /
```

# Go to top directory

Command: `cd /`

```
PS /Users/trh> cd scratch  
PS /Users/trh/scratch> cd /  
PS />
```

# View contents of current directory

Command: **ls**

```
PS /Users/trh> cd scratch  
PS /Users/trh/scratch> cd /  
PS /> ls
```

# View contents of current directory

Command: **ls**

```
PS /Users/trh> cd scratch
PS /Users/trh/scratch> cd /
PS /> ls
Applications          etc
Library               home
Network               installer.failurerequests
System               net
Users                README.txt
PS />
```

# View contents of a file

Command: `cat file-name`

```
PS /Users/trh> cd scratch
PS /Users/trh/scratch> cd /
PS /> ls
Applications          etc
Library               home
Network               installer.failurerequests
System                net
Users                 README.txt
PS /> cat README.txt
```

# View contents of a file

Command: `cat file-name`


```
PS /Users/trh> cd scratch
PS /Users/trh/scratch> cd /
PS /> ls
Applications          etc
Library               home
Network               installer.failurerequests
System               net
Users                README.txt
PS /> cat README.txt
The file says Hello!

PS />
```

# View contents of a file

Command: `cat file-name`

```
PS /Users/trh> cd scratch
PS /Users/trh/scratch> cd /
PS /> ls
Applications          etc
Library               home
Network               installer.failurerequests
System                net
Users                 README.txt
PS /> cat README.txt
The file says Hello!
```



```
PS />
```



# Today's Topics

Terminal Emulators and Shells

Navigation

Running Programs and Commands

- Navigational commands
- Arguments
- Saving output

Demos

# Arguments (program input)

```
PS /Users/trh> cd scratch
PS /Users/trh/scratch> cd /
PS /> ls
Applications          etc
Library               home
Network               installer.failurerequests
System               net
Users                README.txt
PS /> cat README.txt
The file says Hello!

PS />
```

# Arguments (program input)

```
PS /Users/trh> cd scratch
PS /Users/trh/scratch> cd /
PS /> ls
Applications          etc
Library               home
Network              installer failure requests
Users                 README.txt
PS /> cat README.txt
The file says Hello!

PS />
```

program name (cat)

an argument (README.txt)

# echo Example

```
PS /Users/trh>
```

# echo Example

```
PS /Users/trh> echo hello
```

# echo Example

program is "echo"

argument is "hello"

```
PS /Users/trh> echo hello
```

# echo Example

```
PS /Users/trh> echo hello  
hello  
PS /Users/trh>
```

# echo Example

```
PS /Users/trh> echo hello
```

```
hello
```

```
PS /Users>
```

the echo program prints whatever  
it's argument is



# Today's Topics

Terminal Emulators and Shells

Navigation

Running Programs and Commands

- Navigational commands
- Arguments
- Saving output

Demos

# Saving output

Format: `program > file-name`

```
PS /Users/trh> echo hello
```

```
hello
```

```
PS /Users/trh> echo hello > output.txt
```

“redirect” operator, sends output to a file

# Saving output

Format: **program > file-name**

```
PS /Users/trh> echo hello
```

```
hello
```

```
PS /Users/trh> echo hello > output.txt
```

```
PS /Users/trh>
```

# Saving output

Format: **program > file-name**

```
PS /Users/trh> echo hello  
hello
```

```
PS /Users/trh> echo hello > output.txt
```

```
PS /Users/trh>
```

without redirect, output  
was printed to the screen

with redirect, output was  
saved in the output.txt file

# Saving output

Format: `program > file-name`

```
PS /Users/trh> echo hello
hello
PS /Users/trh> echo hello > output.txt
PS /Users/trh>
```

# Saving output

Format: **program > file-name**

```
PS /Users/trh> echo hello
hello
PS /Users/trh> echo hello > output.txt
PS /Users/trh> cat output.txt
```

# Saving output

Format: **program > file-name**

```
PS /Users/trh> echo hello
hello
PS /Users/trh> echo hello > output.txt
PS /Users/trh> cat output.txt
hello
PS /Users/trh>
```

# Today's Topics

Terminal Emulators and Shells

Navigation

Running Programs and Commands

Demos



# Conclusion

Today we covered

- What a terminal and shell is
- What it looks like to have multiple storage drives attached to your computer
- How to navigate between directories/folders
- How to run programs in the terminal