Review: Unix Commands

- What are some commands we can use to check on processes?
- How do we set environment variables?
  - What are examples of environment variables?
- How do we set up aliases?
- Did you customize your prompt?

Today

- Process communication
- Pipes
- Filters

PROCESS COMMUNICATION

Inter-process Communication

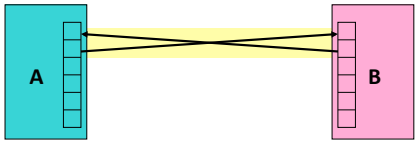
- Ways in which processes communicate:
  - Passing arguments, environment variables
  - Read/write regular files
  - Exit values
  - Signals
  - **Pipes**

One of the cornerstones of UNIX

PIPES

Pipes

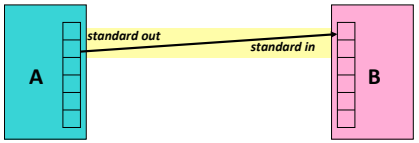
- General idea: The input of one program is the output of the other, and vice versa



- Both programs run at the same time

Pipes

- Often, only one end of the pipe is used



- Could this be done with files/redirection?

Redirection/File Approach

Run first program, save output into file

Run second program, using file as input

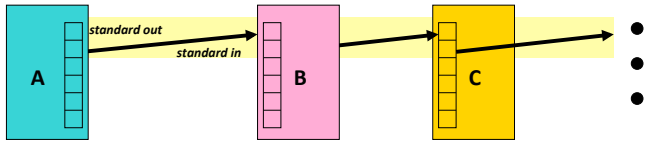
- Unnecessary use of the disk
  - Slower
  - Can take up a lot of space
- Doesn't take advantage of multi-tasking

Coordinating Pipes

- What if a process tries to read data but nothing is available?
  - UNIX puts the reader to sleep until data available
- What if a process can't keep up reading from the process that's writing?
  - UNIX keeps a buffer of unread data
    - This is referred to as the *pipe size*
  - If the pipe fills up, UNIX puts the writer to sleep until the reader frees up space (by doing a read)
- Multiple readers and writers possible with pipes

Filters

- Pipes are often chained together
  - Called *filters*



Pipe Examples

- Output of one program becomes input to another
- Example: `$ who | wc -l`
  - counts the number of users logged in
- Example: `$ find . –mtime 0 | wc`
  - Counts number of files created in last day
- Pipelines can be long:

```
who | awk '{print $1}' | sort | uniq
```


Getting Input: What's the difference?

- Both of these commands send input to *command* from a file instead of the terminal:

```
$ cat file | command
```

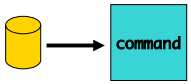
vs.

```
$ command < file
```

Difference: An Extra Process

```
$ cat file | command
```


```
$ command < file
```



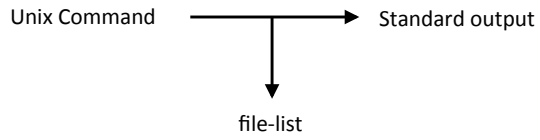
Introduction to Filters

- A class of Unix tools called ***filters***
  - Utilities that read from standard input, transform the input, and write to standard out
- Using filters can be thought of as *data-oriented programming*
  - Each step of the computation transforms data *stream*

Examples of Filters

- `sort`
  - Input: lines from a file
  - Output: lines from the file sorted
- `grep`
  - Input: lines from a file
  - Output: lines that match the argument
- `awk`
  - Programmable filter

`tee`



- Read from standard input and write to standard output and one or more files
  - Captures intermediate results from a filter in the pipeline

tee

- Syntax: `tee [ -ai ] file-list`
  - `-a` append to output file rather than overwrite, default is to overwrite (replace) the output file
  - `-i` ignore interrupts
  - `file-list` one or more file names for capturing output
- Examples

```
$ ls | head –10 | tee first_10 | tail –5
$ who | tee user_list | wc
```

What is the end result of each command?

Unix Text Files: Delimited Data

Tab Separated

```
John    99
Anne    95
Andrew  50
Tim     75
Arun    33
Sowmya  76
```

Lots of other delimiters, e.g., commas or pipes

Why do we use delimiters?

Colon-separated

```
root:x:0:0:root:/root:/bin/bash
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
sgoryl:x:1001:1001:Steve Goryl:/home/faculty/sgoryl:/bin/bash
```


cut: select columns

- `cut` prints selected parts of input lines
  - Can select columns (assumes tab-separated input)
  - Can select a range of character positions
- Some options:
  - `-f listOfCols` print only specified columns (tab-separated) on output
  - `-c listOfPos` print only chars in specified positions
  - `-d c` use character c as the column separator
- Lists are specified as ranges (e.g. 1-5) or comma-separated (e.g. 2,4,5).

cut examples

```
cut -f 1 newdata
cut -f 1-3 newdata
cut –f 4,2 newdata
cut –f 2- newdata
cut –d'@' -f 1 newdata
cut -c 1-4 newdata
```

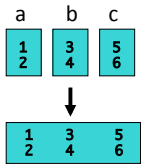
Note how output is formatted
→ Columns joined by delimiter

Unfortunately, there's no way to refer to "last column" without counting the columns.

Get me the list of domains from the email addresses

paste: join columns

- paste displays several text files "in parallel" on output
- If the inputs are files a, b, c
  - the first line of output is composed of the first lines of a, b, c
  - the second line of output is composed of the second lines of a, b, c
- Lines from each file are separated by a tab character
- If files are different lengths, output has all lines from longest file, with empty strings for missing lines



paste example

```
cut -f 1 data > data1
cut -f 2 data > data2
cut -f 3 data > data3
paste data1 data3 data2 > newdata
```

What is each command doing?
What is the final result?

sort: Sort lines of a file

- sort copies input to output but ensures that output is arranged in ascending order of lines.
  - By default, sorting is based on ASCII comparisons of the whole line
- Other features of sort:
  - Understands text data that occurs in columns. (can also sort on a column other than the first)
  - Can distinguish numbers and sort appropriately
  - Can sort files "in place" as well as behaving like a filter
  - Capable of sorting very large files

sort: Options

- `sort [-dftnr] [-o filename] [filename(s)]`

Option	Meaning
-d	Dictionary order, only letters, digits, and whitespace are significant in determining sort order
-f	Ignore case (fold into lower case)
-t	Specify delimiter
-n	Numeric order, sort by arithmetic value instead of first digit
-r	Sort in reverse order
-o	Filename – write output to filename, filename can be the same as one of the input files

- Lots more options…

uniq: list UNIQue items

- Remove or report adjacent duplicate lines
- uniq [-cdu] [input-file] [output-file]
  - -c Supersede the -u and -d options and generate an output report with each line preceded by an occurrence count
  - -d Write only the duplicated lines
  - -u Write only those lines which are not duplicated
  - The default output is the union (combination) of -d and -u

Find all the unique email address domains

wc: Counting results

- The *word count* utility, **wc**, counts the number of lines, characters or words
- Options:
  - -l Count lines
  - -w Count words
  - -c Count characters
- Default: count lines, words and chars

wc and uniq Examples

```
who | sort | uniq –d
wc my_essay
who | wc
sort file | uniq | wc –l
sort file | uniq –d | wc –l
sort file | uniq –u | wc -l
```

Why do we have to execute sort before uniq?
Can't we just use uniq?
(How do you think uniq is implemented?)

yes

- What does this command do?

yes

- Syntax: `yes [STRING]`
- Output a string repeatedly *until killed*