# CS 35L Software Construction Lab

Week 2 – Shell Scripting

## Some more commands

- **cat** – concatenate and print files.
- Examples –
  - cat 01.txt
    to display the contents of file 01.txt.
  - cat 01.txt 02.txt
    to display the contents of both files.
  - cat file1.txt file2.txt > file3.txt
    reads file1.txt and file2.txt and combines those files to make file3.txt.
  - cat note5 >> notes
    attach note5 to notes.
  - cat > file1
    write content to a new file file1 or overwrite file1 if it exists.
  - cat >> file1
    add additional content to file1.

## Some more commands

- **head** - first 10 lines of file, unless otherwise stated.
- Example –
  - head myfile.txt – Would display the first ten lines of myfile.txt.
  - head -15 myfile.txt – Would display the first fifteen lines of myfile.txt.

## Some more commands

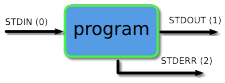
- **tail** - display the last 10 lines of file, unless otherwise stated.
- Example –
  - tail myfile.txt – Would display the last ten lines of myfile.txt.
  - tail -15 myfile.txt – Would display the last fifteen lines of myfile.txt.

## Some more commands

- **grep** - "global regular expression print". It processes text line by line and prints any lines which match a specified pattern.
- Example –
  - ls -l > allFileListing
  - grep csgrad allFileListing
  - Alternatively, ls -l | grep csgrad

## Piping and Redirection

- Every program we run on the command line automatically has three data streams connected to it.
  - STDIN (0) - Standard input (data fed into the program)
  - STDOUT (1) - Standard output (data printed by the program, defaults to the terminal)
  - STDERR (2) - Standard error (for error messages, also defaults to the terminal)




Piping and redirection is the means by which we may connect these streams between programs and files to direct data in useful ways.

## Pipe

- It lets you feed the output from the program on the left as input to the program on the right.
- Example –
  - **ls | head -3**

    barry.txt

    bob

    example.png
  - **ls | head -3 | tail -1**

    example.png

## Redirection (>,>>,<)

- > STDOUT output should be redirected to the file. If the file already exists it will be overwritten.
- >> STDOUT output should be redirected to the file but instead of overwriting, append it to the file if it exists.
- < Read STDIN/input from the file.
- 2> Redirect STDERR to the file specified.

## sort, comm, cmp and tr

- **sort: sorts lines of text files**
  Usage: sort [OPTION]… [FILE]…
- **comm: compare two sorted files line by line**
  Usage: comm [OPTION]… FILE1 FILE2
  comm -23 file1 file2
- **cmp: compare two files byte by byte.** If a difference is found, it reports the byte and line number where the first difference is found. Exit status is 0 if files identical, 1 if different and 2 if problem encountered.
  Usage: cmp [OPTION]… FILE 1 FILE2
- **tr: translate or delete characters**
  Usage: tr [OPTION]…SET1 [SET2]
  echo "password a1b2c3" | tr -d [:digit:] -> password abc
  echo "abc" | tr [:lower:] [:upper:] -> ABC

## sed

**sed: stream editor, modifies the input as specified by the command(s)**

Can be used for:

- Printing specific lines or address ranges
  sed –n '1p' file.txt
  sed –n '1,5p' file.txt
  sed –n '1~2p' file.txt
- Deleting text
  sed '1~2d' file.txt
- Substituting text - s/regex/replacement/flags
  sed 's/cat/dog/' file.txt
  sed 's/cat/dog/g' file.txt
  sed 's/<[^>]*>//g' a.html

## More sed examples

- sed -n 12,18p file.txt
- sed 12,18d file.txt
- sed '1~3d' file.txt
- sed '1,20 s/Johnson/White/g' file.txt
- sed '/pattern/d' file.txt
- sed '/regexp/!d' file.txt

## Regular Expressions

- Quantification
- How many times of previous expression?
- Most common quantifiers: ?(0 or 1), *(0 or more), +(1 or more)

- Alternation
- Which choices?
- Operators: [] and |
  Hello|World [A B C]

- Anchors
- Where?
- Characters: ^ (beginning) and $ (end)

# Regular Expressions

^ start of line
$ end of line
\ turn off special meaning of next character
[ ] match any of enclosed characters, use - for range
[^ ] match any characters except those enclosed in []
. match a single character of any value
* match 0 or more occurrences of preceding character/expression
+ match 1 or more occurrences of preceding

| Expression | Matches |
|---|---|
| **tolstoy** | The seven letters tolstoy, anywhere on a line |
| **^tolstoy** | The seven letters tolstoy, at the beginning of a line |
| **tolstoy$** | The seven letters tolstoy, at the end of a line |
| **^tolstoy$** | A line containing exactly the seven letters tolstoy, and nothing else |
| **[Tt]olstoy** | Either the seven letters Tolstoy, or the seven letters tolstoy, anywhere on a line |
| **tol.toy** | The three letters tol, any character, and the three letters toy. Anywhere on a line |
| **tol.*toy** | The three letters tol, any sequence of zero or more characters, and the three letters toy. Anywhere on a line |

# Quoting - To preserve literal meaning of special characters

- Escape Character \ - Literal value of following character
  echo \|
- Single Quote - Literal Meaning of all within ‘’
  $hello=1
  $str=‘$hello’
  echo $str -> $hello
- Double Quote - Literal meaning except for $, ` and \.
  $hello=1
  $str=“abc$hello”
  echo $str -> abc1

Backquote - execute the command
echo `ls` -> prints result after running ls

# Shell Scripting

- **Shell: The shell provides you with an interface to the UNIX system.**
  -It gathers input from you and executes programs based on that input.
  -When a program finishes executing, it displays that program's output.
- **Shell-script: A file containing shell commands (and comments - preceded by #) to execute**
  - The #! First Line (shebang): a way to tell the kernel which shell to use for a script
  #!/bin/sh
  - Make it executable: chmod +x scriptFile
  - Execute: path_to_script/scriptFile or
  sh path_to_script/scriptFile

# Shell Programming Constructs

**Variables**

- Valid character string [a-zA-Z0-9_] to which a value is assigned
  var_name=var_value !!No spaces around =!!
- Access using $: echo $var_name
- Special Variables: certain characters reserved as special variables
  $: PID of current shell
  #: number of arguments the script was invoked with
  n: nth argument to the script
  ?: exit status of the last command executed
  echo $$; echo $#; echo $2; echo $?;
- scalar variable vs array variable:
  array_name[index]=value; echo ${array_name[index]}

# Accessing Shell Script Arguments

**Example:**

```
$ who | grep betsy                          Where is betsy?
betsy pts/3 Dec 27 11:07
```

**Script:**

```
#! /bin/sh
# finduser --- see if user named by first argument is logged in
who | grep $1
```

**Run it:**

```
$ chmod +x finduser                  Make it executable
$ ./finduser betsy                   Test it: find betsy
betsy pts/3 Dec 27 11:07
$ ./finduser benjamin                Now look for Ben
benjamin dtlocal Dec 27 17:55
```

## Looping

- **for var in list_values**
  **do**
  command 1
  ..
  command n
  **done**

```
ALL=`ls -a $dir | sort`
declare -a ARRAY
count=0
for FILE in $ALL
    do
    ARRAY[$count]=$FILE
    .....
    done
```

```
for i in "${ARRAY[@]}"
    do
    …
    done
```

- **while condition**
  **do**
  command 1
  ..
  command n
  **done**

## Conditional

- if…then…fi
- if…then…else…fi
- if…then…elif..then…fi
- case…esac

```
#!/bin/sh

a=10
b=20

if [ $a == $b ]
then
   echo "a is equal to b"
elif [ $a -gt $b ]
then
   echo "a is greater than b"
elif [ $a -lt $b ]
then
   echo "a is less than b"
else
   echo "None of the condition met"
fi
```

## Unconditional

- break
- continue

```
#!/bin/sh

FRUIT="kiwi"

case "$FRUIT" in
   "apple") echo "Apple pie is quite tasty."
   ;;
   "banana") echo "I like banana nut bread."
   ;;
   "kiwi") echo "New Zealand is famous for kiwi."
   ;;
esac
```

# Lab 2

- Extract lines which contain words (Hint: <td> tag)
- Get lines with Hawaiian words
  - Even numbered lines
- sed 's/<[^>]*>//g' a.html to remove all HTML tags
- Remove leading space
  - sed 's/^\s*//g' (\s denotes space)
- Substitute space in between words to newline
- Delete all commas
- Delete entries which have any character other than Hawaiian
- Sort unique