Lab: Helpful Unix Content

"I searched my .bash_history for the line with the highest ratio of special characters to regular alphanumeric characters, and the winner was:

```bash
cat out.txt | grep -o "\[\[(.\*[])][^\]]\]*$"
```

I have no memory of this and no idea what I was trying to do, but I sure hope it worked."

-- from xkcd #1638
xkcd.com
Slide Credit

• These slides were prepared by S18 TA Brenton Lessley
• However, they are closely derived from slides generated by S17 TA Andrew Hampton.
Standard Streams

• Wikipedia: “preconnected input and output channels between a computer program and its environment (typically a text terminal) when it begins execution”

• Three standard streams:
  – stdin (standard input)
  – stdout (standard output)
  – stderr (standard error)

What mechanisms in C allow you to access standard streams?
printf

• Print to stdout
  – printf(“hello world\n”);
  – printf(“Integers are like this %d\n”, 6);
  – printf(“Two floats: %f, %f”, 3.5, 7.0);
fprintf

• Just like printf, but to streams
• `fprintf(stdout, “helloworld\n”);`
  – → same as printf
• `fprintf(stderr, “helloworld\n”);`
  – prints to “standard error”
• `fprintf(f_out, “helloworld\n”);`
  – prints to the file pointed to by FILE *f_out.
Unix shells allows you to manipulate standard streams.

• “>” redirect output of program to a file

• Example:
  – ls > output
  – echo “this is a file” > output2
  – cat file1 file2 > file3
Unix shells allows you to manipulate standard streams.

- “>>” concatenate output of program to end of existing file
  - (or create file if it doesn’t exist)

- Example:
  - echo “I am starting the file” > file1
  - echo “I am adding to the file” >> file1
  - cat file1
    I am starting the file
    I am adding to the file
Example: redirect output from C program

```c
#include <stdlib.h>
#include <stdio.h>
#include <time.h>

void classify(int log_n, int n){
    int ERROR_NUMBER = 9;
    int got_error = 0;
    int q = n;
    while(q != 0){
        int digit = q % 10;
        q /= 10;
        if(digit == ERROR_NUMBER){
            fprintf(stderr, "log #\%d: \%d\n", log_n, n);
            got_error = 1;
            break;
        }
    }
    if(!got_error){
        printf("log #\%d: \%d\n", log_n, n);
    }
}

int main(void){
    srand(time(0)); // initialize the random number generator
    for(int i=0; i<10000; i++){
        classify(i, rand());
    }
}
```
Example: redirect output from C program

- let's try to write the output of this program to files ... stderr should go to a file called bad_log and stdout should go to a file called good_log

- DEMO
Example: redirect output from C program

• let's try to write the output of this program to files ... stderr should go to a file called bad_log and stdout should go to a file called good_log
  - ./unix_fun 1> good_log 2> bad_log
  - cat good_log bad_log | wc –l
    • Check: should be 10,000 print statements/lines
    • wc: counts number of bytes/words/lines

• stderr and stdout are very useful!
pipes in Unix shells

- represented with “|”
- output of one program becomes input to another program
Unix command: alias

• alias: avoid typing a long command sequence repeatedly

• Example: clean, compile, run
  alias go='make clean; make; ./my_program'

• Example: ssh to ix-dev a lot
  alias ix='ssh username@ix-dev.cs.uoregon.edu'

Put an alias command (or anything else) in your ~/.bashrc file to make it persistent! DEMO
Very useful programs

- grep: keep lines that match pattern, discard lines that don’t match pattern

```
C02LN00GFD58:Documents hank$ ls -l | grep ppt
-rw-r--r--@ 1 hank staff  3278589 Apr  5 11:40 CIS330_Lec2.pptx
-rw-r--r--@ 1 hank staff  2220104 Apr  8 20:57 CIS330_Lec3.pptx
-rw-r--r--@ 1 hank staff  3899863 Jan 21 09:26 CIS610_Lec2.pptx
-rw-r--r--@ 1 hank staff  4629257 Jan 30 10:24 CIS610_Lec3.pptx
-rw-r--r--@ 1 hank staff 21382185 Mar  25 12:40 CIS_colloquium2013.pptx
-rw-r--r--@ 1 hank staff 21382185 Jan  7 12:21 CIS_colloquium_2013.pptx
-rw-r--r--@ 1 hank staff 2172179 Dec  20 15:24 ICS_results.pptx
-rw-r--r--@ 1 hank staff  4841050 Nov 13 10:10 MBTI.pptx
-rw-r--r--@ 1 hank staff  2031749 Apr  5 16:20 SC14_flow.pptx
-rw-r--r--@ 1 hank staff  17972476 Mar 25 12:43 VMV_2013.pptx
-rw-r--r--@ 1 hank staff  98149068 Apr  1 10:25 aachen.pptx
-rw-r--r--@ 1 hank staff  9815146 Feb 24 07:00 childs_poster_SDAV_AHM_2014.pptx
-rw-r--r--@ 1 hank staff   592243 Feb 26 04:09 childs_sdav_slides.pptx
-rw-r--r--@ 1 hank staff  15765504 Feb 13 14:57 cig_exascale.ppt
-rw-r--r--@ 1 hank staff  16699392 Jan  7 12:14 cis610_Lec1.ppt
-rw-r--r--@ 1 hank staff   3159872 Jan  7 11:15 egpgv_cgf.pptx
-rw-r--r--@ 1 hank staff  15767552 Mar  23 02:48 eu_regional_school.ppt
-rw-r--r--@ 1 hank staff  35099136 Mar  25 09:42 eu_regional_school_part1.ppt
-rw-r--r--@ 1 hank staff  10775552 Mar  25 04:49 eu_regional_school_part1B.ppt
-rw-r--r--@ 1 hank staff  72966144 Mar  26 08:43 eu_regional_school_part2.ppt
-rw-r--r--@ 1 hank staff   7571317 Mar  25 12:53 ilm_booth_talk.pptx
```
Very useful programs

• sed: replace pattern 1 with pattern 2
  – sed s/pattern1/pattern2/g
    • s means substitute
    • g means “global” ... every instance on the line

sed is also available in “vi”
:%s/pattern1/pattern2/g (% means all lines)
:103,133s/p1/p2/g (lines 103-133)
Example: difficult output from C program

```c
#include <stdlib.h>
#include <stdio.h>
#include <time.h>

void classify(int log_n, int n){
    int ERROR_NUMBER = 9;
    int got_error = 0;
    int q = n;
    while(q != 0){
        int digit = q % 10;
        q /= 10;
        if(digit == ERROR_NUMBER){
            printf("ERROR log #\%d: %d\n", log_n, n);
            got_error = 1;
            break;
        }
    }
    if(!got_error){
        printf("log #\%d: %d\n", log_n, n);
    }
}

int main(void){
    srand(time(0)); // initialize the random number generator
    for(int i=0; i<10000; i++){
        classify(i, rand());
    }
}
```

"fun2.c" 28L, 496C 1,1 All
Example: difficult output from C program

• This one is harder! All of the output goes to stdout, but we still want to separate it. Use pipes, grep, and sed to sort the output into two files called bad_log and good_log ... in bad_log, the word ERROR should be removed. It's ok to write to an intermediate file and do this in several steps.

• DEMO
Example: difficult output from C program

• Solution #1:
  – ./unix_fun > log
  – grep ERROR log | sed 's/ERROR //g' > bad_log
  – grep –v ERROR log > good_log

• Solution #2:
  – ./unix_fun | tee >(grep ERROR | sed 's/ERROR //g' > bad_log) >(grep -v ERROR > good_log)

• Check: Each line in bad_log should have a ‘9’ digit
  – cat bad_log | grep ‘[9]’ | wc –l (this equals 6025)
  – cat bad_log | wc -l (this also equals 6025 – good!)
Unix command: tee

man tee

Tee - read from standard input and write to standard output and files

SYNOPSIS

```
tee [OPTION]... [FILE]...
```

DESCRIPTION

Copy standard input to each FILE, and also:

- `-a`, `--append`
  append to the given FILEs, do not override

- `-i`, `--ignore-interrupts`
  ignore interrupt signals

- `-p`
  diagnose errors writing to non pipes

- `--output-error[=MODE]`
  set behavior on write error. See MODE

- `--help`
  display this help and exit

- `--version`
  output version information and exit

MODE determines behavior with write errors on the outputs:
  'warn' diagnose errors writing to any output

Manual page `tee(1)` line 1 (press h for help or q to quit)
Unix commands: curl

• QUESTION: What’s the weather going to be like tomorrow in Eugene?
• BUT: You can only use your terminal to find out 😊

• DEMO
Unix commands: curl

• Interact with remote servers, view and parse webpage content ...and many more options!
• Great for downloading files (pdfs, .c/.h files,...) right inside the terminal
  – If I can get the link/url of the file, I always do it this way!
  – Avoid using the “Save As” dialogue box
• Example: curl http://wttr.in/~Eugene
  – What’s it do? Try it out right now!
Example: curl

• Let's download some of the CIS 330 lecture slides and tar just the PDFs
  — DEMO
Example: curl

• let's download some of the CIS 330 lecture slides and tar just the PDFs
  – URL=http://ix.cs.uoregon.edu/~hank/330/lectures
  – curl -O $URL/CIS330_S18_Lec1.pdf
  – curl -O $URL/CIS330_S18_Lec1.pptx
  – curl -O $URL/CIS330_S18_Lec2.pdf
  – curl -O $URL/CIS330_S18_Lec2.pptx
  – tar -cvf lecture_notes.tar *.pdf
Wildcards

• ‘*’ is a wildcard with unix shells

fawcett:tmp child$ ls
Abe    Chavarria        Hebb         Macy     Smith
Alajaji  Chen       Jia         Maguire   Steelhammer
Alamoudi  Clark      Kine        Michlanski  Szczepanski
Anastas  Collier    Lee         Moreno    Totten
Andrade  Costello  Legge       Olson      Vega-Fujioka
Ballarche  Donnelly  Li         Owen      Wang
Brennan  Etzel     Lin         Pogrebinsky  Whiteley
Brockway  Friedrich  Liu         Qin       Woodruff
Brogan    Garvin   Lopes       Rhodes     Xu
Brooks    Gonzales  Luo         Roberts    Yaconelli
Bruce    Guo        Lyon        Rodriguez  Young
Carlton    Hampton  Machado     Roush      Zhang
Chalmers  Harris   Machado     Rozenboim  de

fawcett:tmp child$ ls C*
Carlton    Chavarria  Clark       Costello
Chalmers  Chen     Collier

fawcett:tmp child$ ls *z
Rodriguez

fawcett:tmp child$ ls *ee*
Lee       Steelhammer

fawcett:tmp child$ ls *e*e*
Lee        Legge       Steelhammer  Whiteley

‘?’ is a wildcard that matches exactly one character
Other useful shell things

• ‘tab’: auto-complete
  – almost impossible to communicate how great this is!!!
• esc=: show options for auto-complete
• Ctrl-A: go to beginning of line
• Ctrl-E: go to end of line
• Ctrl-R: search through history for command
• Unix tools: clear (clear off terminal screen), top/htop (monitor CPU resources and processes), ”mv” (rename a folder/file)