Homework 4

Nicholas LaRosa

February 19, 2013

1. (a) i. `sed -n '/Nick.*\"/p' test`
   ii. `grep 'Nick.*\"' test`
   iii. `awk '/Nick.*\"/{print}' test`

   (b) i. `sed -n '/\(\([0-9]\|[0-9]\{2\}\|1[0-9]\{2\}\|2[0-4][0-9]\|25[0-5]\)\.\)\{3\}\([0-9]\|[0-9]\{2\}\|1[0-9]\{2\}\|2[0-4][0-9]\|25[0-5]\)/p' messages`
   ii. `egrep '(([0-9]|[0-9]{2}|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.){3}([0-9]|[0-9]{2}|1[0-9]{2}|2[0-4][0-9]|25[0-5])' messages`
   iii. `awk -v d="([0-9]|[0-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])" '{if($0 ~ d"\\."d"\\."d"\\."d) print}' messages`

2. (a) `sed -e '/printf/ {n}' -e 's/pow/powVariable/g' -e 's/sqrt/sqrtVariable/' main.c`

   (b) `sed -e '/Object Oriented Programming/ { s/Object Oriented Programming/ Object Oriented Programming \(OOP\)/ : loop s/Object Oriented Programming/ OOP/ b loop }' TechReport.txt`

3. (a) `grep -n 'gnarly$' /afs/nd.edu/coursesp.13/cse/cse20189.01/files/hw4/*`

   (b) `grep -vc '^\s*$' main.c`

4. (a) `awk -F, '{if( $1 >= $2 ) { print $1 } else { print $2 }}' numbers.txt`

   (b) `ps -au | awk 'BEGIN { sum = 0 } sum = sum + $3; END{ print "Total CPU usage: " sum "%" }'`

5. (a) `awk 'match($0,/DPT=[0-9]*[^ ] /) {print substr($0,RSTART,RLENGTH)}' messages | awk -F= 'BEGIN {dpt80 = 0; dpt22 =0} {if( $2 == 80 ){dpt80 = dpt80 + 1} else if( $2 == 22 ){dpt22 = dpt22 + 1}}; END {print "\n" dpt22 " connected to port 22\n" dpt80 " connected to port 80\n"}'`

Port Number	Attempts
22	3395
80	4573

   Awk was chosen for this assignment due to its ability to keep track of values through its system of declarable variables. Additionally, awk provides the ability to print formatted strings to the screen. As seen in the above command, awk first finds the occurrences of "DPT=" followed by a number (non-greedy) and prints only the matched segment of the line. The results of this command are then piped to another awk command, which separates fields with the "=" character, allowing us to reference the port number as the awk variable $2. Declaring variables to keep track of the number of port 80 and port 22, awk then increments these variables at every occurrence of port 80 or 22. After all the lines have been passed, the command finally prints the two variables in readable format.

   (b) `awk -f coefs.awk coefs.txt`

```
#!/bin/awk -f
BEGIN {
        FS = ":";       # a:b:c, so $1 = a, $2 = b, $3 = c
        x1 = 0;         # roots x1 and x2
        x2 = 0;
}
{
if( $2^2 - 4*$1*$3 >= 0 ){
        x1 = (-$2 + sqrt( $2^2 - 4*$1*$3 )) / ( 2*$1 );
        x2 = (-$2 - sqrt( $2^2 - 4*$1*$3 )) / ( 2*$1 );
        print "(" x1 ",0):(" x2 ",0)";
}
else{
        print "NaN:NaN";
}
}
```

Awk was again chosen because of its C-like structure and its arithmetic abilities. The script begins with a modification of the field separator FS that allows the variables a, b, and c to be referenced within the script as $1, $2, and $3 respectively. Using an if-conditional loop, the awk script checks for non-imaginary roots at every line, and if a non-negative determinant is present, calculates both roots. If not, the output of the respective line reads "NaN:NaN". The output is printed after every line, so no END statement was necessary in this script.