

CS 2413 Test 1

1. [10] Write a sed command which will print all non-blank comment lines that are in the file in awk and pipe the output into the program wc. The command should be entirely on the command line.

Assume that a comment line always has a '#' in the first column and a blank comment line has only white space after the '#'

2. [15] Your machine has been under attack so you have been running snort to monitor the ethernet traffic for connection attempts which represent possible attacks. Because you are monitoring traffic for a lot of machines you want to write a sed script which will print the lines concerning your machine in a more readable format. Your machine name is nereid and your IP address is 129.115.11.213. The lines that you want to print have your IP address as either the source or the destination and you want to replace your IP address with nereid. The relevant data lines are:

10/20-13:00:50.317946/129.115.237.66:1126 -> 129.115.11.213:8080

10/20-12:53:00.766322/129.115.11.213:63602 -> 129.115.98.10:161

and you want them changed to

10/20 13:00:50 129.115.237.66:1126 -> nereid:8080

10/20 12:53:00 nereid:63602 -> 129.115.98.10:161

3. [15] An ISP (Internet Service Provider) has a data file which contains a month's worth of information about user's connect times and bandwidth usage as well as storage usage. There are two types of lines in the file. a connect line for each user connection which has 5 fields containing the login, start and stop times as well as the bandwidth usage broken into the number of bytes in versus the number of bytes out.

<user name> <start time> <stop time> <bytes in> <bytes out>

maynard 1378

2463

12379894

34563

\$1

\$2

\$3

\$4

\$5

and daily storage usage lines which contains the number of bytes of disk space being used by the user which has the following format:

storage <user name> <bytes of storage>

storage maynard

27846439

\$1

\$2

\$3

Write an awk script to find whether the user maynard has a total connect time exceeding 10.000 or if his total bandwidth in or his total bandwidth out exceeds 10.000.000.000 bytes or whose maximum storage for the month exceeded 10.000.000. If any of these conditions are valid, then print out all statistics for the user maynard. The output should look like:

maynard

Connect Time = 1085

Bandwidth In = 12314457

Bandwidth Out = 98456

Storage = 27846439

500-500 = 0

+= 0

+= 0

\$5

4. [10] Write a perl script which will print out the pathnames of all C files which are under any of the directories whose names are given on the command line.
5. [20] Suppose a perl program, `pgtar`, is invoked with the command:

`pgtar -x -f file1 file2`

and suppose `file1` contains "dir1", "dir2" while `file2` contains "dir3", "dir4" and "dir5" and `file3` contains "dir6" and "dir7". Each directory name in the files is on a separate line. At the start of the program:

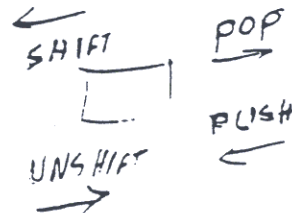
- What is the list value of `@ARGV`?
- What is the value of `$ARGV[1]`?
- What is the value of `$#ARGV`?
- What value is used in the condition test:

`if (@ARGV) {`

- If the following statements are executed first,

```
$a = shift;
$b = pop;
$c = shift;
$d = shift;
$e = <>;
```

- What is the value of `$b`?
- What is the value of `$e`?
- What is the value of `$ARGV`?



- On the other hand if the following statements were to be executed first.

```
$a = shift;
$b = shift;
$c = pop;
push(@ARGV, 'file3');
@c = <>;
$d = pop @c;
```

- What is the value of `$c[1]`?
- What is the value of `@c`?
- What is the value of `$d`?

6. [20] Write a Perl script. `pgrep`, which will search all TEXT files under the directories given on the command line for the regular expression given as the first argument on the command line. Print out the pathname of each file in which a match is found. A pathname should be printed a maximum of once. A sample invocation:
- `pgrep '\w+ [mM]aynard' dir1 dir2 dir3`

CS 2413 Test 1 Solutions

1. [10]

```
sed -n '/~#.*[ ]/p' < in.awk | wc
or
sed '/~#[ ]*/d;/~#[ ]/d' < in.awk | wc
```

print contents of pattern space (not needed as delete command causes a new line of input to be read into the pattern space + the edit restarted)

2. [15]

```
/129.115.11.312/!d
s/-//
s/\.[0-9]* //
s/129.115.11.213/nereid/g
```

This could also have been done by invoking sed with the -n switch and the following script:

```
s/-//
s/\.[0-9]* //
s/129.115.11.213/nereid/g
/nereid/p
```

3. [15]

```
BEGIN{ time=0; in=0; out=0; storage=0 }
$1 == "maynard" {
    time += $3 - $2
    in += $4
    out += $5
}
$1 == "storage" && $2 == "maynard" {
    if ( $3 > storage ) {
        storage = $3
    }
}
END{
    if ( time > 10000 || in > 1000000000000 ||
        out > 1000000000000 || storage > 10000000 ) {
        print "maynard"
        print "\tConnect Time\t=", time
        print "\tBandwidth In\t=", in
        print "\tBandwidth Out\t=", out
        print "\tStorage\t=", storage
    }
}
```

4. [10]

```
print 'find @ARGV -name '*.c' -print';
```

5. [20]

- (a) @ARGV = ('-x', '-f', 'file1', 'file2')
- (b) \$ARGV[2] = '-f'
- (c) \$#ARGV = 3
- (d) scalar @ARGV == 4
- (e)
 - i. \$b = 'file2'
 - ii. \$e = '' (actually what is read from STDIN)
 - iii. \$ARGV == '' (actually \$ARGV = '-')
- (f)
 - i. \$c[1] == "dir2\n"
 - ii. @c = ('dir1\n', 'dir2\n', 'dir6\n')
 - iii. \$d = "dir7\n"

6. [20]

```
$rex = shift;  
@files = 'find @ARGV -print';  
chop @files;  
@tfiles = grep(-f $rex -T, @files);  
foreach $file (@tfiles) {  
    if ( grep ( /$rex/, 'cat $file' ) ) {  
        print $file, "\n";  
    }  
}
```

7. [15]

```
die "Usage: psplit filename\n" unless @ARGV == 1;  
@lines = <>;  
open(OUT1, ">$ARGV[0].1") || die "Could not open $ARGV[0].1 for write: $!\n";  
open(OUT2, ">$ARGV[0].2") || die "Could not open $ARGV[0].2 for write: $!\n";  
print OUT1 @lines[0..$#lines/2];  
print OUT2 @lines[$#lines/2+1..$#lines];
```

7. [15] Write a Perl script, called `psplit`, which will take a `Text` file, whose name is on the command line, and break the file into 2 pieces, each piece essentially the same size, and put the pieces into separate files named the same as the original file but with sequential numbers at the end. Thus the command `psplit head.doc` would create the files `head.doc.1`, `head.doc.2`