Lizhen Lin 82226790

CS146
=====
Assignment #2

(2) 1. Explain what would happen if the following did not give error messages:

```
$ cat x y > y
```

It will end up in an infinite loop. File x remains its content, but file y gets “infinite large” which contains repeated values of file x’s content.

Because the exec(cat) happens after the fork, so since redirect stdout to file y, so first close stdout (fd=1), then create/open file y and assign the lowest available file descriptor which is 1.

After finish the I/O redirect, now to start executing the cat, first try to copy file x’s content to file y, so now file y has file x’s content (might take multiple times depend on the size of file X). Then copy first part of file y’s content and append (because file y remains open after finishing cat x >y) into file y. Since file y is large, so the first cat hasn’t reached EOF yet, and now file y just got larger, so this step will keep repeating, and cat will never reach EOF, so infinite loop.

```
$ cat x >> x
```

It will end up in an infinite loop. File x gets “infinite large”. Because the fork happens before exec(cat), so first close stdout and open file x and assign lowest available file descriptor to file x which is 1. Then start to cat, since it’s the ‘>>’ which means this is the appending mode, and with file x being really large, so it will only be able to copy first part of file x, and appends to end of file x, and then copy second part of file x, and appends to end of file x. Which means this will keep adding more stuff to file x, so cat will never reach EOF, so this process will never terminate, and file x will keep getting bigger which cause an infinite loop.

Lizhen Lin 82226790

(2) 2. Write a short shell script called "cx" that will execute the command "chmod +x" on every file given on its command line. What does it do?

```
    chmod +x $@
```

Turns on the -x bit(executable) for all the files that been passed in.

(2) 3. Write a short shell script called "nf" to display the number of files in the current directory.

```
    ls . | wc -l
```

(ls -a if we also want to count the ignored files)

(4) 4. Write a shell script called "lss" that will list all the files in the current directory in decreasing order of the number of bytes in the file. It does not need to take any arguments. Eg.,

```
$ lss
-rwxrwxr--    1 wayne   faculty      385 Nov 29  1996 lss -
rwxrwxr--    1 wayne   faculty      42 Mar  9  1990 nf
```

```
   ls  -S  -l
```

(-S: sort by size, -l: long format)

(4) 5. Write a shell script called "whoson" to display a sorted list of undergrad students logged in on the current machine. You should take a look at the command called "groups" to acquire some of the requisite information. eg.,

```
         $ whoson
         frank jane jim john laura
```

```
    exec 2>err.txt
    users=(`who -q | head -1`)
    user_arr=( $(printf "%s\n"  "${users[@]}" | sort -u) )
    for usr in "${user_arr[@]}";
```

```
          do
          pattern="${usr} : "
          usr_g=`groups $usr`
          usr_ng=`echo $usr_g | sed "s/$pattern//g"`
          if [[ $usr_ng =~ ugrad ]];then
               printf $usr
               printf " "
          fi
     done
     echo
```

(2) 6. Write a shell script called "howmany" to display the number of undergrad students logged in on the current machine. Use whoson if you can. eg.,

```
         $ howmany
         5
```

```
      exec 2>err.txt
      users=(`who -q | head -1`)
      count=0
      user_arr=( $(printf "%s\n"  "${users[@]}" | sort -u) )
      for usr in "${user_arr[@]}";
          do
          pattern="${usr} : "
          usr_g=`groups $usr`
          usr_ng=`echo $usr_g | sed "s/$pattern//g"`
          if [[ $usr_ng =~ ugrad ]];then
               count=$((count+1))
          fi
      done
      echo $count
```

(4) 7. Write a shell script called "valid" that determines if it's argument is a valid shell variable name, eg

```
         $ valid foobar
         yes
         $ valid 12foobar
         no
```

Lizhen Lin 82226790

```
re1='[0-9]'
para=$1
ch1=${para:0:1}
if [[ $ch1 =~ $re1 ]];then
    echo no
elif [[ $para =~ [^a-zA-Z0-9_] ]]; then
    echo no
else
    echo yes
fi
```

(4) 8. Write a shell script called "prargs" that prints out a numbered list of its arguments in the following format:

```
$ prargs a 'b c'
0: "prargs"
1: "a"
2: "b c”
```

```
count=0
echo $count : \"$0 \"
for arg in "$@";
do
    count=$((count+1))
    echo $count : \"$arg\"
done
```