CSE4034 ADVANCED UNIX PROGRAMMING

PROJECT#1 REPORT

Group Participants:

Serkan AYDIN	**150114053**
Taha Yusuf KÖMÜR	**150114064**

Question 1:

Problem: Reading a text file, and for every word in the file and output words in format of x1.x2.x3.x4

- **uppercase (x1)**
- **lowercase (x2)**
- **digits (x3)**
- **others (x4)**

Sub Problems:

1- Checking for the correct # of arguments

```
_notreadinif=1                                          ## flag to read file

if [ $# -ne 1 ]                                        ## Checking if number of arguments
then                                                   ## If wrong usage, then force to e
    echo "Normal usage is <Q1.sh> <filename> but i'am letting you to keep"
    echo -n 'Enter the file name: '
    read _filename
    _notreadinif=0
fi

if [[ _notreadinif -eq 1 ]]                           ## Check if correct argument
then
    _filename=$1;                                      ## Get the first argument
fi
```

2- Iterating file, line by line

```
while IFS= read -r line || [[ -n "$line" ]]; do     ## Get the line in the
```

3- Iterating line, word by word

```
for word in $line; do                                ## Get the word
```

4- Iterating word, character by character

```
    for (( k=0; k<${#word}; k++ )); do              ## Get the character
```

5- Counting the given character types

```
_uppercase=0
_lowercase=0
_totalnumberofdigits=0
_totalnumberofothercharacters=0
```

```
for (( k=0; k<${#word}; k++ )); do          ## Get the character in the word

    ch=${word:k:1};

    if [[ $ch =~ [A-Z] ]]; then             ## Upper case control
        _uppercase=$((_uppercase + 1))
    elif [[ $ch =~ [a-z] ]]; then           ## Lower case control
        _lowercase=$((_lowercase + 1))
elif [[ $ch =~ [0-9] ]]; then           ## Digit control
        _totalnumberofdigits=$((_totalnumberofdigits + 1))
    else
  _totalnumberofothercharacters=$((_totalnumberofothercharacters + 1))
    fi

done
```

6- Giving output in given format

```
    echo -n $_uppercase.$_lowercase.$_totalnumberofdigits.$_totalnumberofotherchara
done
```

example text
SOMETHING HEY123 tell..meeeverythin##G andth()##1

dasdadk

```
(base) duma@Taha-MacBook-Pro 150114053_150114064_Project1 % ./Q1.sh q1.txt
9.0.0.0 3.0.3.0 1.16.0.4 0.5.1.4
0.7.0.0
```

Question 2:

We wrote a recursive function ***half()*** which constructs process binary three.

- Main function calls half() function
- ***half()*** function decreases depth variable(program gets an argument) and if depth is equal to 0 then function returns else continues.

If parent process executes *half()* function;

- ***half()*** function calls fork() so the left child of the current process is created.
- Parent process informs that the left child has been created.
- ***half()*** function calls fork() so the right child of the current process is created.
- Parent process informs that the right child has been created.
- Then the parent process waits for termination of children.
- Parent process informs that the children have been exited.

If left child process executes *half()* function;

- Position of child will be equalized 2 times parent's position(binary tree property)
- Prints it's pid and parent's pid.
- Calls ***half()*** function to create its own children.
- Lastly, exits with it's position.

If right child process executes *half()* function;

- Position of child will be equalized 2 times parent's position + 1(binary tree property)
- Prints it's pid and parent's pid.
- Calls ***half()*** function to create its own children.
- Lastly, exits with it's position.

The output of our C program is below.

```
serkan@serkan:~/CLionProjects/unix$ ./main.o 3
[1]pid 50046 ppid 48675
[1]pid 50046  created child pid 50047
[2]pid 50047 ppid 50046
[1]pid 50046 created child pid 50048
[3]pid 50048 ppid 50046
[2]pid 50047  created child pid 50049
[4]pid 50049 ppid 50047
[3]pid 50048  created child pid 50050
[2]pid 50047 created child pid 50051
[6]pid 50050 ppid 50048
[5]pid 50051 ppid 50047
[3]pid 50048 created child pid 50052
[7]pid 50052 ppid 50048
[2] left child 50049 of 50047 exited status: 4
[2] right child 50051 of 50047 exited status: 5
[3] left child 50050 of 50048 exited status: 6
[3] right child 50052 of 50048 exited status: 7
[1] left child 50047 of 50046 exited status: 2
[1] right child 50048 of 50046 exited status: 3
```

```
serkan@serkan:~/CLionProjects/unix$ ./main.o 3
[1]pid 50198 ppid 48675
[1]pid 50198  created child pid 50199
[2]pid 50199 ppid 50198
[1]pid 50198 created child pid 50200
[3]pid 50200 ppid 50198
[2]pid 50199  created child pid 50201
[4]pid 50201 ppid 50199
[2]pid 50199 created child pid 50203
[3]pid 50200  created child pid 50202
[3]pid 50200 created child pid 50204
[5]pid 50203 ppid 50199
[6]pid 50202 ppid 50200
[7]pid 50204 ppid 50200
[2] left child 50201 of 50199 exited status: 4
[2] right child 50203 of 50199 exited status: 5
[3] left child 50202 of 50200 exited status: 6
[3] right child 50204 of 50200 exited status: 7
[1] left child 50199 of 50198 exited status: 2
[1] right child 50200 of 50198 exited status: 3
```

```
serkan@serkan:~/CLionProjects/unix$ ./main.o 2
[1]pid 50327 ppid 48675
[1]pid 50327  created child pid 50328
[1]pid 50327 created child pid 50329
[2]pid 50328 ppid 50327
[3]pid 50329 ppid 50327
[1] left child 50328 of 50327 exited status: 2
[1] right child 50329 of 50327 exited status: 3
serkan@serkan:~/CLionProjects/unix$
```

```
serkan@serkan:~/CLionProjects/unix$ ./main.o 2
[1]pid 50286 ppid 48675
[1]pid 50286  created child pid 50287
[1]pid 50286 created child pid 50288
[2]pid 50287 ppid 50286
[3]pid 50288 ppid 50286
[1] left child 50287 of 50286 exited status: 2
[1] right child 50288 of 50286 exited status: 3
```