

**CSE 4034**  
**ADVANCED UNIX PROGRAMMING**  
**Project 1 – Report**  
**(Kevser İLDEŞ – 150116048)**

1)

In first part of the project, firstly I checked whether user provide a filename as argument or not using “\$#” and if he/she does not provide one, I read input from console till Ctrl-D is pressed using “cat” command and write to a file named input. Then make that file as parameter using “set” command.

Then I read the file with “cat \$1” command (\$1 parameter either user provided filename or the one we set) and assign the content to text variable.

Then, after reading whole input and assigning to text, I used a for loop for each word in text and in this loop firstly, I create and initialize corresponding values (upper, lower, digit and other) as 0.

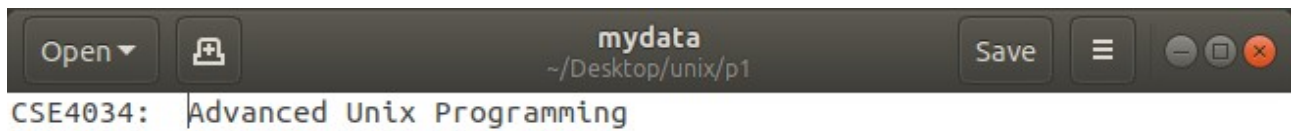
Then I took length of the word using “\${#word}” and till reach to end of the string it takes character in corresponding index with “\${word:\$i:1}” and check whether it is an uppercase, lowercase, digit or any other and increments corresponding variable by 1 (control made with simple if else statements using [A – Z], [a – z] and [0 – 9]).

Next, for each word, after checking all characters it puts numbers in corresponding format and concatenates. As format is x1.x2.x3.x4, this is made by using statement  
“str=\$str”\$upper”.”\$lower”.”\$digit”.”\$other” ””

Then, after searching whole text, finally it prints resulting string to the user using “echo”.

Sample outputs are shown below:

- Content of mydata file:



```
Open ▾  mydata  Save  ~/Desktop/unix/p1
CSE4034: Advanced Unix Programming
```

- Output:



```
kevserildes@kevserildes-Lenovo-IdeaPad-Z500: ~/Desktop/unix/p1
File Edit View Search Terminal Help
kevserildes@kevserildes-Lenovo-IdeaPad-Z500:~$ cd Desktop/unix/p1
kevserildes@kevserildes-Lenovo-IdeaPad-Z500:~/Desktop/unix/p1$ ./unix_1.sh mydata
3.0.4.1 1.7.0.0 1.3.0.0 1.10.0.0
kevserildes@kevserildes-Lenovo-IdeaPad-Z500:~/Desktop/unix/p1$ ./unix_1.sh
CSE4034:
Advanced
Unix
Programming
3.0.4.1 1.7.0.0 1.3.0.0 1.10.0.0
kevserildes@kevserildes-Lenovo-IdeaPad-Z500:~/Desktop/unix/p1$
```

- Content of mydata file:



```
Open ▾  mydata  Save  ~/Desktop/unix/p1
CSE4034: Advanced Unix Programming
Dr. Sanem Arslan Yilmaz
Project1
Kevser-ILDES
150116048
myData saMPle inPut156 file++;
```

- Output:



```
kevserildes@kevserildes-Lenovo-IdeaPad-Z500:~/Desktop/unix/p1$ ./unix_1.sh mydata
3.0.4.1 1.7.0.0 1.3.0.0 1.10.0.0 1.1.0.1 1.4.0.0 1.5.0.0 1.5.0.0 1.6.1.0 6.5.0.1
0.0.9.0 1.5.0.0 2.4.0.0 1.4.3.0 0.4.0.3
kevserildes@kevserildes-Lenovo-IdeaPad-Z500:~/Desktop/unix/p1$ ./unix_1.sh
CSE4034: Advanced Unix Programming
Dr. Sanem Arslan Yilmaz
Project1
Kevser-ILDES
150116048
myData saMPle inPut156 file++;
3.0.4.1 1.7.0.0 1.3.0.0 1.10.0.0 1.1.0.1 1.4.0.0 1.5.0.0 1.5.0.0 1.6.1.0 6.5.0.1
0.0.9.0 1.5.0.0 2.4.0.0 1.4.3.0 0.4.0.3
kevserildes@kevserildes-Lenovo-IdeaPad-Z500:~/Desktop/unix/p1$
```

2)

For the second part of the project, firstly I implemented a function that puts necessary space at the beginning according to corresponding depth value. Then in main function, initializing necessary variables, I took depth input from user and in a while loop, that ends when input is equal to 0, firstly to print info I called indent function and print info to the console.

Next, to construct binary tree, I used a variable named temp that is equal to the depth of the tree and to make only right process to fork I used absolute value of temp-input and compared it with depth of current node. If it is equal, I forked first, left, child and for only parent process using “child1>0” forked again for the right child.

For the child processes, I calculated the traverse number using  $2 * (\text{parent's traverse number})$  for the left child and  $2 * (\text{parent's traverse number}) + 1$  for the right child.

Then, I decrement the input variable. When input is equal to 1 that is when it reaches to leaves of the tree, I broke the loop after printing info.

When loop ends, I called sleep(3) function for all processes to make them wait a bit and then in a while loop, that loops till all children of the process is returned, I took the return value of child as state value using wait(), WIFEXITED() and WEXITSTATUS() calls and then checked whether it is right or left child using returned value, that is traverse number of the child, %2 (if  $\text{trNum} \bmod 2 == 0$  its left child else right). Then after printing information to user, in the end of the code I returned the traverse number of the process.

Here are example outputs of the program:

For depth=3

```
kevserildes@kevserildes-Lenovo-IdeaPad-Z500: ~/Desktop/unix/p1
File Edit View Search Terminal Help
kevserildes@kevserildes-Lenovo-IdeaPad-Z500:~/Desktop/unix/p1$ gcc unix_2.c -o u
nix_2.o
kevserildes@kevserildes-Lenovo-IdeaPad-Z500:~/Desktop/unix/p1$ ./unix_2.o
3
[1] pid 3103, ppid 3088
[1] pid 3103, created left child with pid 3104
[2] pid 3104, ppid 3103
[1] pid 3103, created right child with pid 3105
[2] pid 3104, created left child with pid 3106
[3] pid 3105, ppid 3103
[4] pid 3106, ppid 3104
[2] pid 3104, created right child with pid 3107
[3] pid 3105, created left child with pid 3108
[6] pid 3108, ppid 3105
[3] pid 3105, created right child with pid 3109
[7] pid 3109, ppid 3105
[5] pid 3107, ppid 3104
[2] left child 3106 of 3104 exited with status 4
[3] left child 3108 of 3105 exited with status 6
[3] right child 3109 of 3105 exited with status 7
[2] right child 3107 of 3104 exited with status 5
[1] right child 3105 of 3103 exited with status 3
[1] left child 3104 of 3103 exited with status 2
kevserildes@kevserildes-Lenovo-IdeaPad-Z500:~/Desktop/unix/p1$
```

For depth=5

```
kevserildes@kevserildes-Lenovo-IdeaPad-Z500:~/Desktop/unix/p1$ ./unix_2.o
5
[1] pid 3282, ppid 3261
[1] pid 3282, created left child with pid 3283
[1] pid 3282, created right child with pid 3284
[3] pid 3284, ppid 3282
[2] pid 3283, ppid 3282
[3] pid 3284, created left child with pid 3285
[2] pid 3283, created left child with pid 3286
[3] pid 3284, created right child with pid 3287
[4] pid 3286, ppid 3283
[2] pid 3283, created right child with pid 3288
[6] pid 3285, ppid 3284
[7] pid 3287, ppid 3284
[4] pid 3286, created left child with pid 3289
[6] pid 3285, created left child with pid 3290
[7] pid 3287, created left child with pid 3291
[4] pid 3286, created right child with pid 3292
[6] pid 3285, created right child with pid 3293
[7] pid 3287, created right child with pid 3294
[9] pid 3292, ppid 3286
[8] pid 3289, ppid 3286
[9] pid 3292, created left child with pid 3296
[8] pid 3289, created left child with pid 3295
[8] pid 3289, created right child with pid 3297
[9] pid 3292, created right child with pid 3298
[14] pid 3291, ppid 3287
[15] pid 3294, ppid 3287
[15] pid 3294, created left child with pid 3300
[14] pid 3291, created left child with pid 3299
[14] pid 3291, created right child with pid 3301
[15] pid 3294, created right child with pid 3302
[5] pid 3288, ppid 3283
[17] pid 3297, ppid 3289
[16] pid 3295, ppid 3289
[5] pid 3288, created left child with pid 3303
[29] pid 3301, ppid 3291
```

```
[10] pid 3288, ppid 3287
[5] pid 3288, created left child with pid 3303
[29] pid 3301, ppid 3291
[5] pid 3288, created right child with pid 3304
[18] pid 3296, ppid 3292
[28] pid 3299, ppid 3291
[10] pid 3303, ppid 3288
[31] pid 3302, ppid 3294
[10] pid 3303, created left child with pid 3305
[13] pid 3293, ppid 3285
[13] pid 3293, created left child with pid 3307
[10] pid 3303, created right child with pid 3306
[11] pid 3304, ppid 3288
[13] pid 3293, created right child with pid 3308
[19] pid 3298, ppid 3292
[11] pid 3304, created left child with pid 3309
[30] pid 3300, ppid 3294
[11] pid 3304, created right child with pid 3310
[20] pid 3305, ppid 3303
[12] pid 3290, ppid 3285
[21] pid 3306, ppid 3303
[12] pid 3290, created left child with pid 3311
[26] pid 3307, ppid 3293
[12] pid 3290, created right child with pid 3312
[24] pid 3311, ppid 3290
[27] pid 3308, ppid 3293
[25] pid 3312, ppid 3290
[23] pid 3310, ppid 3304
[22] pid 3309, ppid 3304
[8] right child 3297 of 3289 exited with status 17
[8] left child 3295 of 3289 exited with status 16
[14] right child 3301 of 3291 exited with status 29
[14] left child 3299 of 3291 exited with status 28
[4] left child 3289 of 3286 exited with status 8
[15] right child 3302 of 3294 exited with status 31
[7] left child 3291 of 3287 exited with status 14
[9] left child 3296 of 3292 exited with status 18
[9] right child 3298 of 3292 exited with status 19
```



```

[9] left child 3296 of 3292 exited with status 18
[9] right child 3298 of 3292 exited with status 19
[15] left child 3300 of 3294 exited with status 30
[4] right child 3292 of 3286 exited with status 9
[7] right child 3294 of 3287 exited with status 15
[3] right child 3287 of 3284 exited with status 7
[2] left child 3286 of 3283 exited with status 4
[13] left child 3307 of 3293 exited with status 26
[13] right child 3308 of 3293 exited with status 27
[11] right child 3310 of 3304 exited with status 23
[6] right child 3293 of 3285 exited with status 13
[10] left child 3305 of 3303 exited with status 20
[10] right child 3306 of 3303 exited with status 21
[11] left child 3309 of 3304 exited with status 22
[5] left child 3303 of 3288 exited with status 10
[5] right child 3304 of 3288 exited with status 11
[2] right child 3288 of 3283 exited with status 5
[12] left child 3311 of 3290 exited with status 24
[12] right child 3312 of 3290 exited with status 25
[1] left child 3283 of 3282 exited with status 2
[6] left child 3290 of 3285 exited with status 12
[3] left child 3285 of 3284 exited with status 6
[1] right child 3284 of 3282 exited with status 3

```

→ As mentioned in readme, to make process ids ordered via traverseNum, I used sleep function with traverseNum value for all children and make each creation in time (in ordered code).

Here outputs of this ordered one are provided for some cases:  
For 3:

```

kevserildes@kevserildes-Lenovo-IdeaPad-Z500:~/Desktop/unix/p1$ ./unix_2_ordered.o
3
[1] pid 2163, ppid 2108
[1] pid 2163, created left child with pid 2164
[1] pid 2163, created right child with pid 2165
[2] pid 2164, ppid 2163
[2] pid 2164, created left child with pid 2166
[2] pid 2164, created right child with pid 2167
[3] pid 2165, ppid 2163
[3] pid 2165, created left child with pid 2168
[3] pid 2165, created right child with pid 2169
[4] pid 2166, ppid 2164
[5] pid 2167, ppid 2164
[6] pid 2168, ppid 2165
[2] left child 2166 of 2164 exited with status 4
[7] pid 2169, ppid 2165
[2] right child 2167 of 2164 exited with status 5
[1] left child 2164 of 2163 exited with status 2
[3] left child 2168 of 2165 exited with status 6
[3] right child 2169 of 2165 exited with status 7
[1] right child 2165 of 2163 exited with status 3

```

For 5:

```
kevserildes@kevserildes-Lenovo-IdeaPad-Z500:~/Desktop/unix/p1$ ./unix_2_ordered.o
5
[1] pid 2337, ppid 2108
[1] pid 2337, created left child with pid 2338
[1] pid 2337, created right child with pid 2339
[2] pid 2338, ppid 2337
[2] pid 2338, created left child with pid 2340
[2] pid 2338, created right child with pid 2341
[3] pid 2339, ppid 2337
[3] pid 2339, created left child with pid 2342
[3] pid 2339, created right child with pid 2343
[4] pid 2340, ppid 2338
[4] pid 2340, created left child with pid 2344
[4] pid 2340, created right child with pid 2345
[5] pid 2341, ppid 2338
[5] pid 2341, created left child with pid 2346
[5] pid 2341, created right child with pid 2347
[6] pid 2342, ppid 2339
[6] pid 2342, created left child with pid 2349
[6] pid 2342, created right child with pid 2350
[7] pid 2343, ppid 2339
[7] pid 2343, created left child with pid 2351
[7] pid 2343, created right child with pid 2352
[8] pid 2344, ppid 2340
[8] pid 2344, created left child with pid 2356
[8] pid 2344, created right child with pid 2357
[9] pid 2345, ppid 2340
[9] pid 2345, created left child with pid 2358
[9] pid 2345, created right child with pid 2359
[10] pid 2346, ppid 2341
[10] pid 2346, created left child with pid 2360
[10] pid 2346, created right child with pid 2361
[11] pid 2347, ppid 2341
[11] pid 2347, created left child with pid 2362
[11] pid 2347, created right child with pid 2363
[12] pid 2349, ppid 2342
[12] pid 2349, created left child with pid 2364
```



```
[12] pid 2349, created right child with pid 2365
[13] pid 2350, ppid 2342
[13] pid 2350, created left child with pid 2366
[13] pid 2350, created right child with pid 2367
[14] pid 2351, ppid 2343
[14] pid 2351, created left child with pid 2368
[14] pid 2351, created right child with pid 2369
[15] pid 2352, ppid 2343
[15] pid 2352, created left child with pid 2370
[15] pid 2352, created right child with pid 2371
[16] pid 2356, ppid 2344
[17] pid 2357, ppid 2344
[18] pid 2358, ppid 2345
[8] left child 2356 of 2344 exited with status 16
[19] pid 2359, ppid 2345
[8] right child 2357 of 2344 exited with status 17
[4] left child 2344 of 2340 exited with status 8
[9] left child 2358 of 2345 exited with status 18
[20] pid 2360, ppid 2346
[9] right child 2359 of 2345 exited with status 19
[4] right child 2345 of 2340 exited with status 9
[2] left child 2340 of 2338 exited with status 4
[21] pid 2361, ppid 2346
[22] pid 2362, ppid 2347
[10] left child 2360 of 2346 exited with status 20
[23] pid 2363, ppid 2347
[10] right child 2361 of 2346 exited with status 21
[5] left child 2346 of 2341 exited with status 10
[11] left child 2362 of 2347 exited with status 22
[11] right child 2363 of 2347 exited with status 23
[5] right child 2347 of 2341 exited with status 11
[2] right child 2341 of 2338 exited with status 5
[1] left child 2338 of 2337 exited with status 2
[24] pid 2364, ppid 2349
[25] pid 2365, ppid 2349
[26] pid 2366, ppid 2350
[12] left child 2364 of 2349 exited with status 24
```

```
[27] pid 2367, ppid 2350
[12] right child 2365 of 2349 exited with status 25
[6] left child 2349 of 2342 exited with status 12
[13] left child 2366 of 2350 exited with status 26
[28] pid 2368, ppid 2351
[13] right child 2367 of 2350 exited with status 27
[6] right child 2350 of 2342 exited with status 13
[3] left child 2342 of 2339 exited with status 6
[29] pid 2369, ppid 2351
[30] pid 2370, ppid 2352
[14] left child 2368 of 2351 exited with status 28
[31] pid 2371, ppid 2352
[14] right child 2369 of 2351 exited with status 29
[7] left child 2351 of 2343 exited with status 14
[15] left child 2370 of 2352 exited with status 30
[15] right child 2371 of 2352 exited with status 31
[7] right child 2352 of 2343 exited with status 15
[3] right child 2343 of 2339 exited with status 7
[1] right child 2339 of 2337 exited with status 3
```