CSC3320 System Level Programming
Homework 6

Due at 11:59 pm on Dec. 11th, 2014

Write a C program named “countC.c” taking input text file as arguments and count the characters for that text file according to the following requirements. (If no text file is given, countC will read the input from default text file “test.txt”). You need to use fork to duplicate the processes. The parent process needs to output the most frequent letter and display how many times that letter appears. And please create n child processes (n>=3) and assign each process a task according to the index i (i is from 0 to n-1 but NOT PID). If i%3==0, it should output the number of digits in the text file. If i%3==1, it should output the number of letters in the text file(IGNORE CASE). If i%3==2, it should output the number of white spaces in the text file. (For details, please refer to the example attached)

Below are sample runs of countC.c (assuming countC is the generated executable file):

```
$cat test.txt
This is a list of courses.
CSC 1010 - COMPUTERS & APPLICATIONS
$.countC
Most frequent letter is 's', it appeared 8 times
0---Number of digit = 4
2---Number of space = 12
1---Number of letters = 44
```

Notes:
• 3 child processes are created in this example, and ‘0’, ‘1’, ‘2’ are the indexes of the processes.
• Ignore the case when counting the letters.
• Try to create 10 processes and see what is the output.
• In your output, the order of those indexes may be different from this example.

Submission:
• Upload an electronic copy (MS word or pdf) of your answer sheet (including the source code in your shell scripts and screenshots of outputs) to the folder named “HW6” of the dropbox in the desire2learn system.
• Name your file in the format of HW6_FirstnameLastname (eg. HW6_YuanLong.docx, HW6_YuanLong.pdf).
• Please add the homework number and your name at the top of your answer sheet.
• Upload the source code of “countC.c” to the desire2learn system. And please write your name as a comment at the first line in the file.

Appendix:

E.g.1. Counting occurring frequency

```c
#include <stdio.h>
int mostfrequent(int * a, int length) /*length is the length of the array. The function returns the index of the largest value of array a.*/
{
    int index=0;
    int max=a[0];
    int i;
    for(i=1;i<length;i++)
        if (a[i]>max)
        {
            max = a[i];
            index = i;
        }
    return index;
}
int main() {
    char c;
    int i, nwhite, nother;
    int ndigit[10];
    nwhite = nother = 0;
    for (i=0; i<10; ++i)
        ndigit[i] = 0;
    while ((c = getchar()) != EOF)
    {
        if (c >= '0' && c <= '9')
            ++ndigit[c-'0'];
        else if (c == ' ' || c == '\n' || c == '\t')
            ++nwhite;
        else
            ++nother;
    printf("digits = ");
    for (i=0; i<10; ++i)
        printf("%d ",ndigit[i]);
    printf("\n", white space = %d, other = %d\n",nwhite, nother);
    printf("The most frequent digit is %c\n",'0'+mostfrequent(ndigit,10));
}
E.g.2. Process and file read and write.

```c
#include <unistd.h>
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main(int argc, char * argv[]) {
    int fd;
    char c;
    int i;
    int pid;
    char str[1000];
    if(argc>1) /*argc is the number of command line parameters.*/
        fd=open(argv[1], O_RDWR, 0); /*argv[1] is the first command line parameter.*/
    else
        fd=open("test.txt", O_RDWR, 0); /*if no command line parameter, open file input.txt*/
    if (fd==-1)
    {
        printf("Open file failed.\n");
        return(0);
    }
    else
    {
        while (read(fd,&c,1))
        {
            printf("I read a letter '\%c'\n",c);
        }
    }
    return(1);
}```
E.g.3. Creating multiple child processes

```c
#include <stdio.h>
int main () {
    int pid,i;
    for ( i=0; i<10; i++) {
        pid=fork();
        if (pid==0||pid==-1) {
            break;
        }
    }
    if (pid==-1) {
        printf("error\n");
    } else if(pid==0){
        printf("%d---",i);
        switch (i%3) {
            case 0:
                printf("I am child process A \n");
                break;
            case 1:
                printf("I am child process B \n");
                break;
            case 2:
                printf("I am child process C \n");
                break;
        }
    } else {
        printf("I am parent process\n");
    }
}
```