

## COSC 350 System Software Midterm #1-1

10/06/2021

Name: Jung An

1.

(5 pt.)

`int st_to_int(char *str)`

```
int num = 0;
int i;
for (i = 0; str[i] != '\0'; i++)
    num = (num * 10) + str[i] - '0';
return num;
```

(10 pt.)

void int\_to\_st(char \*str, int num)

```

int sizeCheck = num;
int size = 0;
int i;
for (i = 0; sizeCheck > 0; i++)
{
    sizeCheck /= 10;
    size++;
}
char buffer[size];
sizeCheck = num;
for (i = size - 1; i >= 0; i--)
{
    int nTemp = sizeCheck % 10;
    buffer[i] = nTemp + '0';
    sizeCheck /= 10;
}
str = buffer;

```

you cannot do this

2. (5 pt.) (your answer)

open my.txt from current directory with write only and create file with read and write permission for user

exit program with 1 if there is error opening the file

Print on console: How are you?

dup fd to standard output 1, and exit the program if error dup'ing.

Print: I am fine Thank you How about yourself?

close the file my.txt

exit program

to where?!

## 3. (10 pt.)

```

#include <stdio.h>
#include <stdlib.h>

int st_to_int(char *str)
{
    int sum = 0;
    int neg = 0;
    int i = 0;
    if (str[0] == '-')
    {
        neg = 1;
        i++;
    }
    for (i; str[i] != '\0'; i++)
        sum = (sum * 10) + str[i] - '0';
    if (neg == 1)
        sum = sum * -1;
    return sum;
}

int main (int argc, char **argv)
{
    if (argc == 1)
    {
        printf("Need at least one integer\n");
        return 1;
    }
    int total = 0;
    int i;
    for (i = 1; i < argc; i++)
    {
        int num = st_to_int(argv[i]);
        if (num % 2 == 0)
            total += num;
    }
    printf("The sum of even arguments is %i", num);
    return 0;
}

```

4. (10 pt.)

```
#!/bin/sh
if [ $# -eq 1 ]; then
    echo "Need at least 1 integer"
    exit 1
fi
sum=0
for i in $@
do
    if [ `expr $i % 2` -eq 0 ]; then
        let "sum=$sum+$i"
    fi
done
echo "The sum of even arguments is "$sum
exit 0
```

5. (5 pt.) (Your answer)

Text editor - Editor to write and edit codes

preprocessor - processing the header files before compiler process

compiler - convert the code to machine language

linker - links the header files to corresponding files.

6. (5 pt.) (Your Answer)

a.

\$W

b.

K

## COSC 350 System Software Midterm #1-2

10/09/2020

Name: Jung An7. (5 pt.) What will be the permission for files **foo** and **bar** with following program?

```

#include <unistd.h>
#include <fcntl.h>
#include <ctype.h>

int main ()
{
    umask(0200);
    if (creat("foo", S_IRUSR | S_IWUSR | S_IRGRP | S_IWGRP | S_IROTH | S_IWOTH) < 0)
        return 1;
    umask(0440);
    if (creat("bar", S_IRUSR | S_IWUSR | S_IRGRP | S_IWGRP | S_IROTH | S_IWOTH) < 0)
        return 1;
    return 0;
}

```

Answer) foo = r--rw-rw- ✓bar = -w--w-rw- ✓

8. (10 pt.) Write C code which pass input (text file) and output file name as command line arguments. Open the input file as read only and open output file with mode rw-rw-rw. Your program encodes each character to ASCII code number and writes to output file. You need consider a space and end of line. You need consider argument number error and open file error. You must not use any library function to convert a character to ASCII number. (tip1: use dup2 and type coercion) (tip2: end of line has ASCII number 10).

ex)

input file

output file

AA BB
CC DD

65 65 32 66 66
67 67 32 68 68

```

#include <unistd.h>
#include <fcntl.h>
#include <stdlib.h>
#include <stdio.h>

#define BUFFER_SIZE 1
int main(int argc, char **argv)
{
    if (argc != 3)
    {
        write(2, "Need Two files\n", 15);
        return 1;
    }

    char *i_Name = argv[1];
    char *o_Name = argv[2];
    int f_in, f_out;
    char buffer[BUFFER_SIZE];
    int nread;
    if ((f_in = open(i_Name, O_RDONLY)) == -1)
    {
        write(2, "open error\n", 11);
        return 1;
    }
    if ((f_out = open(o_Name, O_WRONLY | O_CREAT, 0666)) == -1)
    {
        write(2, "open error\n", 11);
        return 1;
    }

    while ((nread = read(f_in, buffer, BUFFER_SIZE)) > 0)
    {
        int ascii = buffer[0] - '\0';
        int temp = ascii;
        int count = 0;
        while (temp > 0)
        {
            temp /= 10;
            count++;
        }
        char ascStr[count];
        int i;
        for (i = count - 1; i >= 0; i--)
        {
            int digit = ascii % 10;
            ascStr[i] = digit + '0';
            ascii /= 10;
        }
        write(f_out, ascStr, count);
        write(f_out, "\n", 1);
    }

    return 0;
}

```

9. (10 pt.) Write a C program that takes two command-line arguments: input file name and output file names. This program read input from the input file and writes in the output file in reverse order without any numerical characters. Created output file mode will be rw-rw-rw-. You need consider argument number error and an input file error.

```

#include <unistd.h>
#include <fcntl.h>
#include <stdlib.h>
#include <stdio.h>
#include <sys/stat.h>

#define BUFFER_SIZE 1

int main (int argc, char** argv)
{
    if (argc != 3)
    {
        write(2, "Need only two files\n", 20);
        return 1;
    }

    char *i_Name = argv[1];
    char *o_Name = argv[2];
    int f_in, f_out;
    char buffer[BUFFER_SIZE];
    int nread;
    if ((f_in = open(i_Name, O_RDONLY)) == -1)
    {
        write(2, "open error\n", 11);
        return 2;
    }
    umask(0000);
    if ((f_out = open(o_Name, O_WRONLY | O_CREAT, 0666)) == -1)
    {
        write(2, "open error\n", 11);
        return 2;
    }

    int offset = lseek(f_in, -1, SEEK_END);

    while (offset >= 0)
    {
        nread = read(f_in, buffer, BUFFER_SIZE);
        if (buffer[0] < '0' || buffer[0] > '9')
            write(f_out, buffer, nread);
        lseek(f_in, -2, SEEK_CUR);
        offset--;
    }

    return 0;
}

```



10. (10 pt.) Write a bash script which checks each of file's type in the current directory and
- If a file is c or c++ program file, compile
  - If a file is text file (.txt), display content of text file on the screen.
  - Other files: just display "file is not c or c++ or text file" on the screen
- Use for loop and case statement (Do not use ()).

*Q1*  
-3

```
#!/bin/sh
for file in *
do
  case $file in
    *.c)      exec gcc $file
    *.cpp)    exec g++ $file
    *.txt)    exec cat $file
    ??)      echo "file is not c or c++ or text file"
  esac
done
exit 0
```

*OK*  
*OK*  
Wrong syntax

11. (5 pt.) The following is the output of command "ls -l"

```
drwxr-xr-x  2 separk  users          512 Sep 16 12:47 csc350/
lrwxr-xr-x  1 separk  users          01233 Sep 22 2011 some1
-rw-----  1 separk  users          43008 Jan  6  2003 snap.doc
```

Answer the following questions:

- 1) Explain the meaning of "d", "-" and "l" at the beginning of the two lines.

*-15*  
d means directory

l means library file

- 2) What are the permissions for group for csc350?

read and execute

- 3) Give a command to add executable permission to others for snap.doc.

chmod o+x

- 4) Give a command to remove read permission for group for csc350.

chmod g-r



12. Look following source codes. Let's assume all files are located in directory /home/separk. Write command or commands for each question.

```
/* File foo.c */
#include <stdio.h>
#include "BF.h"
int main()
{
    printf("%s\n", bill());
    printf("%s\n", fred());
    return 0;
}
```

```
/* File BF.h */
#ifndef BF_H
#define BF_H
char *bill();
char *fred();
#endif
```

```
/* File Fred.c */
char * fred()
{
    return "fred";
}
```

```
/* File Bill.c */
char * bill()
{
    return "bill";
}
```

- (1 pt.) Create foo.o, Fred.o and Bill.o.

*gcc foo.c  
gcc Fred.c  
gcc Bill.c*

- (3 pt.) Create a static library named libBF.a that contains Bill.o and Fred.o.

*gcc Bill.o Fred.o -c libBF.a*

- (1 pt.) Move libBF.a to /home/separk/bin

*mv libBF.a ~/bin*

- (1 pt.) Use gcc to create an executable file named foo by linking foo.o with the libBF.o from the library in /home/separk/bin

*gcc foo.o ~/bin/libBF.o -o foo*

13. (4 pt.) Why do we call that system calls are unbuffered I/O and library functions are buffered I/O?

*System call operator does not have to go through other parts to be called while library functions has to go through many different parts such as system call to be used*

