CMPS 401
Survey of Programming Languages

Programming Assignment #3
C Language
On the Ubuntu Operating System

1. Write a small C program (P3.c) to roll your own **shell** that loops reading a line from standard input and checks the first word of the input line. If the first word is one of the following internal commands (or aliases) perform the designated task. Otherwise use the standard ANSI C system function to execute the line through the default system shell.

Internal Commands/Aliases:

myclear

Clear the screen using the system function clear: system("clear")
The string "clear" is passed to the default command shell for execution.

mydir <directory>

List the directory contents (ls -al <directory>)
You will need to provide some command line parsing capability to extract the target directory for listing. Once you have built the replacement command line, use the system function to execute it.

myenviron

List all the environment strings
The environment strings can be accessed from within a program as a global variable.
    extern char **environ;
The environ is an array of pointers to the environment strings terminated with a NULL pointer.

myquit

Quit from the program with a zero return value. Use the standard exit function.

External Commands:

For all other command line inputs, relay the command line to the parent shell for execution using the system function (i.e. who).

Sample Run:

```
==>mydir
total 28
drwxrwxr-x 2 yang yang 4096 Sep 24 02:56 .
drwxrwxr-x 6 yang yang 4096 Sep 24 01:45 ..
-rwxrwxr-x 1 yang yang 8975 Sep 24 02:56 a.out
-rw-rw-r-- 1 yang yang 4340 Sep 24 02:55 P3.c
==>cal
   September 2013
Su Mo Tu We Th Fr Sa
 1  2  3  4  5  6  7
 8  9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30
==>myquit
```

2. You need a CMPS401 account on the Ubuntu Operating System.
3. If you need help with this assignment, go to the following links:

   C Programming Language Tutorials / Reference

   - C Programming (book from Wikibooks)
   - C Language Tutorial (book by Gordon Dodrill)
   - The C Programming Language 2nd Ed (book by Brian Kernighan and Dennis Ritchie)
   - UNIX Tutorial

4. The following instructions are used to edit, compile, and run your program

```
•  nano      P3.c     // Ctrl + z: Exit the editor
•  gcc       P3.c     // gcc compiler generates default executable file: a.out
•  ./a.out            // Run your executable file
```

5. There are 12 C programming examples to assist you on this assignment.
   1) A simple program to run (TSimple.c)
   2) Test data types and variables (TVar.c)
   3) Other concerns: Test Pointers (TPointer.c)
   4) Test selection statements (TSel.c)
   5) Test loops (TLoop.c)
   6) Other concerns: Test Struct and Union (TStructUnion.c)
   7) Test subprograms (TSub.c)
   8) Other concerns: Test Scope (TScope.c)
   9) Other concerns: Test Scanf (TScanf.c)
   10) Other concerns: Test System (TSystem.c)
   11) Other concerns: Test Environment (TEnviron.c)
   12) Other concerns: Test Strtok (TStrtok.c)

6. Your program assignment #3 consists of the following two files under your CMPS401 account **home directory**:
   1) C program (P3.c)
   2) Executable file (a.out)

   Note: Your files on the Ubuntu Operating System will be checked and should not be modified after due date.

Useful UNIX Commands

Command	**Description**
man	help menu
pico	simple text editor
gcc	compiles your source code “gnu C compiler”
a.out	executes your program
ls –al	displays a long list of files “includes hidden files i.e. dot files”
pwd	prints working directory “pathname”
cd	changes directory
mkdir	creates a directory
rmdir	removes a directory
cp file1 file2	copies contents of file1 into file2
mv file1 file2	moves a file from one place to another, or change its name
rm	removes a file
more	displays a file’s contents
grep	searches for a specified pattern in a file or list of files
ps	obtains the status of the active processes in the system
kill –9 pid	terminates all processes
passwd	modify a user's password
logout	terminates your session
who	display who is on the system
finger	displays the user information
date > myfile	“output redirection” saves the output of date command in myfile
cal >> myfile	“appends” calendar to myfile
cal	display a calendar and the date
wc file1	counts the number of lines, words, and characters in file1

Examples

A Simple Program to Run (TSimple.c)

```
// Display "Hello" on your screen
// Program-ID:   TSimple.c
// Author:       Kuo-pao Yang
// OS:           Ubuntu 20
// Compiler:     GNU C
// Note:
// The following instructions are used to
//      edit, compile, and run this program
//    $ nano     TSimple.c
//    $ gcc      TSimple.c
//    $ ./a.out

#include <stdio.h>

void main() {
   printf("Hello\n");
}

/* Output
    Hello
*/
```

Data Types and Variables (TVar.c)

```c
// Test Data types and variables
// int, float, double, char, char[](String)
// NO boolean variable in C
// Program-ID:   TVar.c
// Author:       Kuo-pao Yang
// OS:           Ubuntu 20
// Compiler:     GNU C
// Note:
// The following instructions are used to
//      edit, compile, and run this program
//    $ nano     TVar.c
//    $ gcc      TVar.c
//    $ ./a.out

#include <stdio.h>

void main() {
    int i1 = 1, i2 = 2;
    float f1 = 3.3, f2 = 4.4;
    double d = 5.5;
    char c = 'a';
    char s[] = "bcd";
    char* t = s;

    f1 = i1 ; // implicit casting
    i2 = f2;  // type narrowing (no type checking in C)
    printf("i1 = %d\n", i1); //%d: decimal format
    printf("f1 = %f\n", f1); //%f: floating point format
    printf("d  = %lf\n",d ); //%lf: double floating point format
    printf("f2 = %f\n", f2);
    printf("i2 = %d\n", i2);
    printf("c  = %c\n", c ); //%c: char
    printf("s  = %s\n", s ); //%s: string
    printf("t  = %s\n", t );
}

/* Output:
    i1 = 1
    f1 = 1.000000
    d  = 5.500000
    f2 = 4.400000
    i2 = 4
    c  = a
    s  = bcd
    t  = bcd
*/
```

Other concerns: Test Pointers (TPointer.c)

```
// Test Pointer
// Program-ID:   TPointer.c
// Author:       Kuo-pao Yang
// OS:           Ubuntu 20
// Compiler:     GNU C
// Note:
// The following instructions are used to
//      edit, compile, and run this program
//    $ nano     TPointer.c
//    $ gcc      TPointer.c
//    $ ./a.out

#include <stdio.h>

void main() {
    int n1, n2, n3;    //declare var names
    int *p1, *p2, *p3; //declare pointers to integers

    p1 = &n1;          //assign "address of" n1 to p1
    p2 = &n2;          //assign "address of" n2 to p2
    p3 = &n3;          //assign "address of" n3 to p3

    //Put values into memory locations pointed to by the ptrs
    n1  = 5;
    *p2 = 7;           // assign 7 to "deference" p2
    *p3 = *p1 + *p2;

    //Print out the adresses of the vars and their contents
    printf("Address\t\t\t Content\t\t Dereference\n");
    printf("-------\t\t\t -------\t\t -----------\n");
    printf("&n1 = %p\t n1 = %d\n", &n1, n1); //%p: address of pointer
    printf("&n2 = %p\t n2 = %d\n", &n2, n2); //\t: tab, \n: new line
    printf("&n3 = %p\t n3 = %d\n", &n3, n3); //%d: decimal format
    printf("&p1 = %p\t p1 = %p\t *p1 = %d\n", &p1, p1, *p1);
    printf("&p2 = %p\t p2 = %p\t *p2 = %d\n", &p2, p2, *p2);
    printf("&p3 = %p\t p3 = %p\t *p3 = %d\n", &p3, p3, *p3);
}

/* Output
Address                  Content                Dereference
-------                  -------                -----------
&n1 = 0x7ffeb481e834     n1 = 5
&n2 = 0x7ffeb481e838     n2 = 7
&n3 = 0x7ffeb481e83c     n3 = 12
&p1 = 0x7ffeb481e840     p1 = 0x7ffeb481e834    *p1 = 5
&p2 = 0x7ffeb481e848     p2 = 0x7ffeb481e838    *p2 = 7
&p3 = 0x7ffeb481e850     p3 = 0x7ffeb481e83c    *p3 = 12
*/
```

Selection Statements (TSel.c)

```c
// Test Selections:      if, if-else, nested if-else
// Logical Operators:    &&, ||, !
// Relational Operators: <, >, ==, <=, >=, !=
// Program-ID:   TSel.c
// Author:       Kuo-pao Yang
// OS:           Ubuntu 20
// Compiler:     GNU C
// Note:
// The following instructions are used to
//      edit, compile, and run this program
//    > pico     TSel.c
//    > gcc      TSel.c //default executable: a.out
//    > a.out

#include <stdio.h>

void main() {
   int i1=1, i2=2, i3=3, i4=4, i5=5, i6=6;

   // Test a simple if
   if (i4 > i1) printf("i4 >  i1\n");

   // Test if-else
   if ((i5 < i2) && (i3 >= i2))
      printf("(i5 <  i2) && (i3 >= i2)\n");
   else
      printf("(i5 >= i2) || (i3 <  i2)\n");

   // Test nested if-else
   if (i1 != i2) {
      printf("(i1 != i2)\n");
   }
   else {
      if ((i4 == i5) || (i5 != i6)) {
         printf("(i1 == i2)&& ((i4 == i5) || (i5 != i6))");
      }
   }
}

/* Output:
i4 >  i1
(i5 >= i2) || (i3 <  i2)
(i1 != i2)
*/
```

Loops (TLoop.c)

```c
// Test Loop: while, for, nested loops (1-D Arrary and 2-D Array)
// Program-ID:   TLoop.c
// Author:       Kuo-pao Yang
// OS:           Ubuntu 20
// Compiler:     GNU C
// Note:
// The following instructions are used to
//      edit, compile, and run this program
//    $ nano     TLoop.c
//    $ gcc      TLoop.c
//    $ ./a.out

#include <stdio.h>

void main() {
    int i, j;
    int a[3] = {1, 2, 3};
    int b[3][3] = {{10, 20, 30},
                   {40, 50, 60},
                   {70, 80, 90}};
    int *p;

    printf("\nTest while loop: 1-D Array and Pointer\n");
    p = &a[0];
    i = 0;
    while(i < 3) {
       printf("a[%d] = %d, *p = %d\t", i, a[i], *p);
       p++;
       i++;
    }

    printf("\nTest for loop: 2-D Array and Pointer\n");
    p = &b[1][0];
    for(j = 0; j < 3; j++) {
        printf("b[1,%d] = %d, *p = %d\t", j, b[1][j], *p);
        p++;
    }

    printf("\nTest nested loop: 2-D Array and Pointer");
    p = &b[0][0];
    for(i = 0; i < 3; i++) {
      printf("\n");
      for(j = 0; j < 3; j++) {
        printf("b[%d,%d] = %d, *p = %d\t", i, j, b[i][j], *p);
        p++;
      }
    }
}

/* Output:
Test while loop: 1-D Array and Pointer
a[0] = 1, *p = 1        a[1] = 2, *p = 2        a[2] = 3, *p = 3
Test for loop: 2-D Array and Pointer
b[1,0] = 40, *p = 40    b[1,1] = 50, *p = 50    b[1,2] = 60, *p = 60
Test nested loop: 2-D Array and Pointer
b[0,0] = 10, *p = 10    b[0,1] = 20, *p = 20    b[0,2] = 30, *p = 30
b[1,0] = 40, *p = 40    b[1,1] = 50, *p = 50    b[1,2] = 60, *p = 60
b[2,0] = 70, *p = 70    b[2,1] = 80, *p = 80    b[2,2] = 90, *p = 90
*/
```

Other concerns: Test Struct and Union (TStructUnion.c)

```
// Test Struct and Union
// Program-ID:   TStructUnion.c
// Author:       Kuo-pao Yang
// OS:           Ubuntu 20
// Compiler:     GNU C
// Note:
// The following instructions are used to
//      edit, compile, and run this program
//    $ nano     TStructUnion.c
//    $ gcc      TStructUnion.c
//    $ ./a.out

#include <stdio.h>

struct date {
    int y;
    int m;
    int d;
};

union u {
    int i;
    int j;
};

void main() {
    struct date b[2];
    b[0].y = 1980; b[0].m = 10; b[0].d = 12;
    b[1].y = 1986; b[1].m = 11; b[1].d = 22;
   struct date *p;
    p = b;
    printf("Test Struct and Pointer\n");
    printf("b[0].y = %d\t (*p).y = %d\t p->y = %d\n", b[0].y,(*p).y, p->y);
    printf("b[0].m = %d\t (*p).m = %d\t p->m = %d\n", b[0].m,(*p).m, p->m);
    printf("b[0].d = %d\t (*p).d = %d\t p->d = %d\n", b[0].d,(*p).d, p->d);
    p++;
    printf("b[1].y = %d\t (*p).y = %d\t p->y = %d\n", b[1].y,(*p).y, p->y);
    printf("b[1].m = %d\t (*p).m = %d\t p->m = %d\n", b[1].m,(*p).m, p->m);
    printf("b[1].d = %d\t (*p).d = %d\t p->d = %d\n", b[1].d,(*p).d, p->d);

    union u q;
    union u *r = &q;
    q.i = 20;
    q.j = 30;
    printf("Test Union and Pointer\n");
    printf("q.i = %d\t (*r).i = %d\t r->i = %d\n", q.i,(*r).i, r->i);
    printf("q.j = %d\t (*r).j = %d\t r->j = %d\n", q.j,(*r).j, r->j);
}

/* Output
Test Struct and Pointer
b[0].y = 1980    (*p).y = 1980   p->y = 1980
b[0].m = 10      (*p).m = 10     p->m = 10
b[0].d = 12      (*p).d = 12     p->d = 12
b[1].y = 1986    (*p).y = 1986   p->y = 1986
b[1].m = 11      (*p).m = 11     p->m = 11
b[1].d = 22      (*p).d = 22     p->d = 22
Test Union and Pointer
q.i = 30         (*r).i = 30     r->i = 30
q.j = 30         (*r).j = 30     r->j = 30
*/
```

Subprograms (TSub.c)

```c
// Test Subprograms: Call by Value and Call by Reference
// Program-ID:   TSub.c
// Author:       Kuo-pao Yang
// OS:           Ubuntu 20
// Compiler:     GNU C
// Note:
// The following instructions are used to
//      edit, compile, and run this program
//    $ nano     TSub.c
//    $ gcc      TSub.c
//    $ ./a.out

#include <stdio.h>

int func1(int i1, int *j1) {
    i1 = i1 + 1;
    *j1 = *j1 + 2;
    printf("func1() i1 = %d, j1 = %d\n", i1, *j1);
    return (i1 + *j1);
}
void func2(int i2, int j2[]) {
    i2 = i2 + 3;
    j2[0] = j2[1] + 4;
    printf("func2() i2 = %d, j2 = %d\n", i2, j2[0]);
}
void func3(int i3, int *j3){
    i3 = i3 + 3;
    *j3 = *(j3 + 1) + 4;
    printf("func3() i3 = %d, j3 = %d\n", i3, *j3);
}

void main() {
    //Test call by value and call by reference
    printf("Test call by value and call by reference\n");
    int n1 = 1, n2 = 2;
    int n3 = func1(n1, &n2);
    printf("n1 = %d, n2 = %d, n3 = %d\n", n1, n2, n3);

    //Test Array to Subprogram
    int i;
    int a[3] = {10, 20, 30};
    printf("Test Array to Subprogram: way 1 (array)\n");
    func2(a[1], a);
    for(i = 0; i < 3; i++) {
       printf("a[%d] = %d\t", i, a[i]);
    }
    printf("\nTest Array to Subprogram: way 2 (pointer, same result)\n");
    a[0] = 10; a[1] = 20; a[2] = 30;
    func3(a[1], a);
    for(i = 0; i < 3; i++) {
       printf("a[%d] = %d\t", i, a[i]);
    }
}

/* Output:
Test call by value and call by reference
func1() i1 = 2, j1 = 4
n1 = 1, n2 = 4, n3 = 6
Test Array to Subprogram: way 1 (array)
func2() i2 = 23, j2 = 24
a[0] = 24       a[1] = 20       a[2] = 30
Test Array to Subprogram: way 2 (pointer, same result)
func3() i3 = 23, j3 = 24
a[0] = 24       a[1] = 20       a[2] = 30
*/
```

Other concerns: Test Scope (TScope.c)

```
// Test Scope: global, local, block, and static variables
// Program-ID:   TScope.c
// Author:       Kuo-pao Yang
// OS:           Ubuntu 20
// Compiler:     GNU C
// Note:
// The following instructions are used to
//      edit, compile, and run this program
//    $ nano     TScope.c
//    $ gcc      TScope.c
//    $ ./a.out

#include <stdio.h>

int i = 1;       //global var i

void func() {
    printf("func() global i = %d\n", i);
    static int j = 4; //static var j
    j++;
    printf("func() static j = %d\n", j);
}

void main() {
   int i = 2;   //local var i
    {
      int i = 3; //block var i
      printf("block  i = %d\n", i);
    }
    printf("main() i = %d\n", i);
    func();
   func();
}

/* Output:
block  i = 3
main() i = 2
func() global i = 1
func() static j = 5
func() global i = 1
func() static j = 6
*/
```

Other concerns: Test Scanf (TScanf.c)

```
// Test Data types and variables
// int, float, double, char, char[](String)
// NO boolean variable in C
// Program-ID:   TScanf.c
// Author:       Kuo-pao Yang
// OS:           Ubuntu 20
// Compiler:     GNU C
// Note:
// The following instructions are used to
//      edit, compile, and run this program
//    $ nano     TScanf.c
//    $ gcc      TScanf.c
//    $ ./a.out

#include <stdio.h>

void main() {
    int i;
    printf("Enter integer i: ");
    scanf("%d", &i); //%d: decimal format, &: address of
    printf("i = %d\n", i );

    float f;
    printf("Enter float f: ");
    scanf("%f", &f); //%f: float, &: address of
    printf("f = %f\n", f );

    double d;
    printf("Enter double d: ");
    scanf("%lf", &d); //%f: float, &: address of
    printf("d = %lf\n", d );

    char s[80];
    printf("Enter string s[80]: ");
    scanf("%s", s); //%s: string, NO address of (&)
    printf("s = %s\n", s );
    printf("s+1 = %s\n", s+1 );
}

/* Output:
Enter integer i: 1
i = 1
Enter float f: 2.2
f = 2.200000
Enter double d: 3.3
d = 3.300000
Enter string s[80]: abcd
s = abcd
s+1 = bcd
*/
```

Other concerns: Test System Function (TSystem.c)

```
// Test System Function
//    system: pass a command to the shell
// Program-ID:   TSystem.c
// Author:       Kuo-pao Yang
// OS:           Ubuntu 20
// Compiler:     GNU C
// Note:
// The following instructions are used to
//      edit, compile, and run this program
//    $ nano     TSystem.c
//    $ gcc      TSystem.c
//    $ ./a.out

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

void main() {
   system("ls -al");
}

/* Output:
drwxrwxr-x 2 yang yang 4096 Sep 24 02:35 .
drwxrwxr-x 6 yang yang 4096 Sep 24 01:45 ..
-rwxrwxr-x 1 yang yang 8381 Sep 24 02:35 a.out
-rw-rw-r-- 1 yang yang 2243 Aug 28 20:55 TEnviron.c
-rw-rw-r-- 1 yang yang 1662 Sep 24 02:28 TLoop.c
-rw-rw-r-- 1 yang yang 1747 Sep 24 02:05 TPointer.c
-rw-rw-r-- 1 yang yang 1145 Aug 28 20:55 TScanf.c
-rw-rw-r-- 1 yang yang  856 Aug 28 20:55 TScope.c
-rw-rw-r-- 1 yang yang 1030 Aug 28 20:55 TSel.c
-rw-rw-r-- 1 yang yang  407 Aug 28 20:55 TSimple.c
-rw-rw-r-- 1 yang yang  965 Aug 28 20:55 TStrtok.c
-rw-rw-r-- 1 yang yang 1797 Aug 28 20:55 TStructUnion.c
-rw-rw-r-- 1 yang yang 1800 Sep 24 02:33 TSub.c
-rw-rw-r-- 1 yang yang 1239 Aug 28 20:55 TSystem.c
-rw-rw-r-- 1 yang yang 1137 Aug 28 20:55 TVar.c
*/
```

Other concerns: Test Environment (TEnviron.c)

```
// Test Environment
// Program-ID:   TEnviron.c
// Author:       Kuo-pao Yang
// OS:           Ubuntu 20
// Compiler:     GNU C
// Note:
// The following instructions are used to
//      edit, compile, and run this program
//    $ nano     TEnviron.c
//    $ gcc      TEnviron.c
//    $ ./a.out

#include <stdio.h>

extern char **environ;        // environment array

void main() {
   char ** env = environ;
   while (*env)
      printf("%s\n",*env++);  // step through environment
}

/* Output:
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd=40;33;01
:or=40;31;01:mi=00:su=37;41:sg=30;43:ca=30;41:tw=30;42:ow=34;42:st=37;44:ex=01;32:*.tar=0
1;31:*.tgz=01;31:*.arc=01;31:*.arj=01;31:*.taz=01;31:*.lha=01;31:*.lz4=01;31:*.lzh=01;31:
*.lzma=01;31:*.tlz=01;31:*.txz=01;31:*.tzo=01;31:*.t7z=01;31:*.zip=01;31:*.z=01;31:*.Z=01
;31:*.dz=01;31:*.gz=01;31:*.lrz=01;31:*.lz=01;31:*.lzo=01;31:*.xz=01;31:*.zst=01;31:*.tzs
t=01;31:*.bz2=01;31:*.bz=01;31:*.tbz=01;31:*.tbz2=01;31:*.tz=01;31:*.deb=01;31:*.rpm=01;3
1:*.jar=01;31:*.war=01;31:*.ear=01;31:*.sar=01;31:*.rar=01;31:*.alz=01;31:*.ace=01;31:*.z
oo=01;31:*.cpio=01;31:*.7z=01;31:*.rz=01;31:*.cab=01;31:*.wim=01;31:*.swm=01;31:*.dwm=01;
31:*.esd=01;31:*.jpg=01;35:*.jpeg=01;35:*.mjpg=01;35:*.mjpeg=01;35:*.gif=01;35:*.bmp=01;3
5:*.pbm=01;35:*.pgm=01;35:*.ppm=01;35:*.tga=01;35:*.xbm=01;35:*.xpm=01;35:*.tif=01;35:*.t
iff=01;35:*.png=01;35:*.svg=01;35:*.svgz=01;35:*.mng=01;35:*.pcx=01;35:*.mov=01;35:*.mpg=
01;35:*.mpeg=01;35:*.m2v=01;35:*.mkv=01;35:*.webm=01;35:*.ogm=01;35:*.mp4=01;35:*.m4v=01;
35:*.mp4v=01;35:*.vob=01;35:*.qt=01;35:*.nuv=01;35:*.wmv=01;35:*.asf=01;35:*.rm=01;35:*.r
mvb=01;35:*.flc=01;35:*.avi=01;35:*.fli=01;35:*.flv=01;35:*.gl=01;35:*.dl=01;35:*.xcf=01;
35:*.xwd=01;35:*.yuv=01;35:*.cgm=01;35:*.emf=01;35:*.ogv=01;35:*.ogx=01;35:*.aac=00;36:*.
au=00;36:*.flac=00;36:*.m4a=00;36:*.mid=00;36:*.midi=00;36:*.mka=00;36:*.mp3=00;36:*.mpc=
00;36:*.ogg=00;36:*.ra=00;36:*.wav=00;36:*.oga=00;36:*.opus=00;36:*.spx=00;36:*.xspf=00;3
6:
SSH_CONNECTION=147.174.34.161 61026 147.174.34.21 4422
LESSCLOSE=/usr/bin/lesspipe %s %s
LANG=en_US.UTF-8
XDG_SESSION_ID=10
USER=yang
PWD=/home/yang/public_html/Cmps401/P3C/Examples
HOME=/home/yang
SSH_CLIENT=147.174.34.161 61026 4422
XDG_DATA_DIRS=/usr/local/share:/usr/share:/var/lib/snapd/desktop
SSH_TTY=/dev/pts/0
MAIL=/var/mail/yang
TERM=xterm
SHELL=/bin/bash
SHLVL=1
LOGNAME=yang
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus
XDG_RUNTIME_DIR=/run/user/1000
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/g
ames:/snap/bin
LESSOPEN=| /usr/bin/lesspipe %s
_=./a.out
*/
```

Other concerns: Test String trtok (TStrtok.c)

```
// Test Strtok Function
//    Split a string into tokens by sparators (delimiters)
// Program-ID:   TStrtok.c
// Author:       Kuo-pao Yang
// OS:           Ubuntu 20
// Compiler:     GNU C
// Note:
// The following instructions are used to
//      edit, compile, and run this program
//    $ nano     TStrtok.c
//    $ gcc      TStrtok.c
//    $ ./a.out

#include <string.h>
#include <stdio.h>

#define MAX_BUFFER   1024              // max line buffer
#define MAX_ARGS       64              // max # args
#define SEPARATORS   " \t\n"           // token sparators

void main() {
   char  cmd[MAX_BUFFER] = "ls -al";
   char* args[MAX_ARGS];               // pointers to arg strings
   int   i;

   args[0] = strtok(cmd, SEPARATORS);  // tokenize input
   printf("args[0] = %s\n", args[0]);
   for (i = 0; args[i] = strtok(NULL, SEPARATORS); i++)
      printf("args[%d] = %s\n",i, args[i]);
}

/* Output:
args[0] = ls
args[1] = -al
*/
```