Grupo CSI-81		1151018	Sistemas Operativos
Documento	Proyecto final		
Fecha	7-XII-18		
Trimestre	T-18O		
Actividad: **SIMULADOR DE UNA HERRAMIENTA DE ADMINISTRACIÓN DE UNIX**			
Calificación S			

```
/*********************************************************
*                                *
*     Header donde se declaran los prototipos y      *
*     las cabeceras principales                  *
*                                *
*                                *
*                                *
*                                *
*********************************************************/
#include <time.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/wait.h>
#include <fcntl.h>
#include <errno.h>
#define MAXARG 512
#define MAXBUF 512
enum plano{PRIMERPLANO,SEGUNDOPLANO};
typedef enum {FALSO,CIERTO} bool;
enum tokens{EOL=1,ARG,AMPERSAND,PUNTOYCOMA};
bool esarguento(char);
int sistema(char **,int);
void procesalinea(char *);
```

```
int dametoken(char **,char *,char *,char **,char **);
int leerlinea(char *,char *);
void instanciasN(int opc);
void erroresB(char *);
void comandosB(char *);
void imprimeB(char *);

#!/bin/sh
#Script para compilar la libreria
ACTUAL=`pwd`
cd ../../lib
LIB=`pwd`
cd $ACTUAL
cd ../incl
INCL=`pwd`
cd $ACTUAL
make $* RUTA_LIB=$LIB/ RUTA_INCL=$INCL/
#Scrip para el apoyo de la compilacion
ACTUAL=$PWD
cd ../../bin
BIN=`pwd`
cd $ACTUAL
cd ../../lib
LIB=`pwd`
cd $ACTUAL
cd ../incl
INCL=`pwd`
cd $ACTUAL
make $* RUTA_BIN=$BIN/ RUTA_LIB=$LIB/ RUTA_INCL=$INCL/
```

```
/*******************************************************
*                                     *
*      Funcion que se encarga de generar bitacora de   *
*      errores                              *
*                                     *
*                                     *
*                                     *
*                                     *
*******************************************************/
#include "headeruamashell.h"
void comandosB(char *bufferentrada)
{
      FILE *archivo;
      archivo = fopen("../fuente/util/uamashell.log","a");
      fprintf(archivo,"Comando: %s",bufferentrada);
      fclose(archivo);

}

/*******************************************************
*                                     *
*      Funcion que se encarga de tokens individuales   *
*      de la linea de comandos                    *
*                                     *
*                                     *
*                                     *
*                                     *
*******************************************************/
```

```
#include "headeruamashell.h"

int dametoken(char **apuntadorsalida,char *bufferentrada,char *buffertoken,char
**apuntador,char **token)

{
	int tipo;
	*apuntadorsalida=*token;
	/* Saltarse los blancos */
	for(;**apuntador==' ' || **apuntador=='\t';(*apuntador)++);
	*(*token)++=**apuntador;
	switch(*(*apuntador)++)
	{
		case '\n':
			tipo=EOL;
		break;
		case '&':
			tipo=AMPERSAND;
		break;
		case ';':
			tipo=PUNTOYCOMA;
		break;
		default:
			tipo=ARG;
			while(esarguento(**apuntador))
			*(*token)++=*(*apuntador)++;
		break;
	}
	*(*token)++='\0';
	return(tipo);
}
```

```
/*******************************************************
*                                     *
*      Funcion que se encarga de generar bitacora de   *
*      errores                              *
*                                     *
*                                     *
*                                     *
*                                     *
*******************************************************/
#include "headeruamashell.h"
void erroresB(char error[])
{
       time_t t;
      struct tm *tm;
      char fechayhora[100];

      t=time(NULL);
      tm=localtime(&t);
      strftime(fechayhora, 100,  "%H:%M.%S, %A de %B de %Y", tm);
      FILE *archivo;
      archivo = fopen("../fuente/util/uamashell_error.log","a");
      fprintf(archivo,"Error: %s, FechaHora: %s \n",error,fechayhora);
      fclose(archivo);

}

/*******************************************************
*                                     *
*      Funcion que determina si un caracter forma      *
```

```
*      parte de un argumento del comando a ejecutar    *
*                                      *
*                                      *
*                                      *
*                                      *
*******************************************************/
#include "headeruamashell.h"
bool esarguento(char c)
{
      char *trabajo;
      char especial[]={' ','\t','&',';','\n','\0'};
      for(trabajo=especial;*trabajo;*trabajo++)
      if(c==*trabajo)
      return(FALSO);
      return(CIERTO);
}

/*******************************************************
*                                      *
*      Funcion que se encarga de imprimir              *
*      cualquier archivo que se mande como argumento   *
*                                      *
*                                      *
*                                      *
*                                      *
*******************************************************/

#include "headeruamashell.h"
```

```
void imprimeB(char *archivo) {
	FILE *bitacoraC;
	bitacoraC = fopen(archivo, "r");
	char line[512];
	int contador = 0;

	while (fgets(line, sizeof(line), bitacoraC)) {
		printf("%s", line);
		contador++;

		if (contador == 22) {
		printf("Press Enter to continue...");

		while (getchar() != '\n');
		contador = 0;
		}
	}

	fclose(bitacoraC);
	return;
}

/*******************************************************
*                                     *
*      Funcion que se encarga de saber cuantas        *
*      instancias se abrieron                         *
*                                     *
*                                     *
*                                     *
```

```
*                                   *
*****************************************************/
#include "headeruamashell.h"
void instanciasN(int opc)
{
    int N;
    if(opc==0)//Se abre una instancia
    {
        FILE *archivo;
        archivo = fopen("../fuente/util/instancias.log","r");
        if (archivo == NULL)
        {
            printf("\n-Error-\n");
            erroresB("Error en la apertura del archivo instancias.log");
        }
        else
        {
            N=(unsigned int)fgetc(archivo)-48;
        }
        fclose(archivo);
        archivo = fopen ( "../fuente/util/instancias.log", "w" );
        fprintf(archivo,"%d",N+1);
        fclose(archivo);
    }
    if(opc==1)
    {
        FILE *archivo;
        archivo = fopen("../fuente/util/instancias.log","r");
        if (archivo == NULL)
```

```
            {
                printf("\nError de apertura del archivo. \n\n");
            }
            else
            {
                N=(unsigned int)fgetc(archivo)-48;
            }
            fclose(archivo);
            archivo = fopen ( "../fuente/util/instancias.log", "w" );
            fprintf(archivo,"%d",N-1);
            fclose(archivo);

        }

}

/*********************************************************
*                                   *
*       Funcion que se encarga leer la entrada del      *
*       usuario en por teclado                  *
*                                   *
*                                   *
*                                   *
*                                   *
*********************************************************/
#include "headeruamashell.h"
int leerlinea(char *prompt,char *bufferentrada)
{
        int caracter,contador;
```

```
    /* Mostrar prompt*/
    printf("%s ",prompt);

    for(contador=0;;)
    {
        if((caracter=getchar())==EOF)
        return (EOF);
        if(contador<MAXBUF)
        bufferentrada[contador++]=caracter;

        if(caracter=='\n' && contador<MAXBUF)
        {
            bufferentrada[contador]='\0';
            erroresB("Error, la linea es muy larga");
            return(contador);
        }
        /* Si la linea es muy larga resetear */
        if(caracter=='\n')
        {
            contador=0;
            printf("%s ",prompt);
        }

    }


}

/*******************************************************
```

```
*                                   *
*      Funcion que se encarga del parseo de la linea   *
*      usando dametoken                        *
*                                   *
*                                   *
*                                   *
*                                   *
*****************************************************/
#include "headeruamashell.h"
void procesalinea(char *bufferentrada)
{
      char *arg[MAXARG+1];
      char buffertoken[2*MAXBUF];
      char *apuntador=bufferentrada;
      char *token=buffertoken;
      int tipotoken;
      int contador;
      int tipo;
      comandosB(bufferentrada);
      for(contador=0;;)
      {
            /* Actuar conforme al tipo de tokenen */

switch(tipotoken=dametoken(&arg[contador],bufferentrada,buffertoken,&apuntador,&token))
            {
            case ARG:
                  if(contador<MAXARG)
                  contador++;
            break;
```

```
            case EOL:
            case PUNTOYCOMA:
            case AMPERSAND:
                tipo=(tipotoken==AMPERSAND)?
                PRIMERPLANO:SEGUNDOPLANO;
                if(contador)
                {
                    arg[contador]=NULL;
                    sistema(arg,tipo);
                }
                if(tipotoken==EOL)
                    return;
                contador=0;
            break;
            }
        }
}

/*******************************************************
*                                  *
*      Funcion permite utilizar las herramientas      *
*      del shell para poder hacer la simulacion       *
*                                  *
*                                  *
*                                  *
*                                  *
*******************************************************/
#include "headeruamashell.h"
int sistema(char **lineacomandos,int donde)
```

```
{
	pid_t pid,estadosalida,retorno;
	if((pid=fork())<0)
	{
		erroresB("Error al crear proceso adicional para shell");
		error("Shell interrupcion");
	}
	if(!pid)
	{
		execvp(*lineacomandos,lineacomandos);
		error(*lineacomandos);
	}


	/* Espera hasta que el hijo termine */
	while((retorno=wait(&estadosalida))!=pid && retorno !=-1);
		return((retorno==-1)?-1:(estadosalida>>8,estadosalida&=0xFF));
}

#include <stdio.h>
#include <stdlib.h>
int main(int argc, char** argv)
{

	FILE *archivo;
char error[10]="Hola";
	archivo = fopen("BitacoraErrores.log","a");
	fprintf(archivo,"Error: %s\n",error);
	fclose(archivo);
```

```
	return 0;


}
/*********************************************************
*                                       *
*      Funcion principal que se encarga de iniciar     *
*      el shell                                 *
*                                       *
*                                       *
*                                       *
*                                       *
*********************************************************/
#include "headeruamashell.h"
int main(int argc,char **argv)
{
	char *prompt;
	char bufferentrada[MAXBUF];
	instanciasN(0);
	prompt="\nUAM_Azc_Shell> ";
	while(leerlinea(prompt,bufferentrada)!=EOF)
	{
		if(strcmp(bufferentrada,"terminar\n")==0)
		{
			instanciasN(1);
			break;
		}
		if(strcmp(bufferentrada,"bitacora_error\n")==0)
		{
			imprimeB("../fuente/util/uamashell_error.log");
```

```
            }
            if(strcmp(bufferentrada,"bitacora_comandos\n")==0)
            {

                imprimeB("../fuente/util/uamashell.log");
            }
            if(strcmp(bufferentrada,"ayuda\n")==0)
            {

                imprimeB("../fuente/util/ayuda.log");
            }



            procesalinea(bufferentrada);
        }
        return(0);
}

# Archivo de descripcion para poder generar la libreria
LIBRERO=ar rcs
COMPILADOR=gcc
FUENTE = dametoken.c esargumento.c leerlinea.c procesalinea.c sistema.c instanciasN.c erroresB.c
comandosB.c imprimeB.c
OBJS=$(FUENTE:.c=.o)
HEADER=$(RUTA_INCL)headeruamashell.h
LIB= $(RUTA_LIB)libUAMShellLibreria.a
C_LIB:  $(OBJS)
        $(LIBRERO) $(LIB) $(OBJS)
        @- echo "Compilacion de la libreria terminada"
```

```
LIMPIA:
	@- rm -f $(OBJS)
	@- echo "Borrado de objetos terminada"
BORRA:
	@- rm -f $(OBJS) $(LIB)
	@- echo "Borrado de archivos terminada"
.c.o:
	$(COMPILADOR) -c -I$(RUTA_INCL) $*.c
$(OBJS): $(HEADER)
# %W%
COMPILADOR=cc
FUENTE = uamashell.c
OBJS=$(FUENTE:.c=.o)
HEADER=headeruamashell.h
LIB=UAMShellLibreria
PROG= $(RUTA_BIN)UAM
CC: $(OBJS)
	$(COMPILADOR) $(OBJS) -L$(RUTA_LIB) -l$(LIB) -lm -o $(PROG)
	@- echo "Compilacion terminada"
LIMPIA:
	@- rm -f $(OBJS)
	@- echo "Borrado de objetos terminado"
BORRA:
	@- rm -f $(OBJS) $(PROG)
	@- echo "Borrado de archivos terminado"
.c.o:
	$(COMPILADOR) -c -I$(RUTA_INCL) $*.c -o $*.o
$(OBJS):$(RUTA_INCL)$(HEADER)  $(RUTA_LIB)lib$(LIB).a
```