

Trabajo Práctico N° 0: Infraestructura Básica

Martinez Ariel, *Padrón Nro. 88573*
lamc1986@yahoo.com.ar

Ali Luis, *Padrón Nro. 81151*
aliluis@gmail.com

Dworjanyn Enrique, *Padrón Nro. 84253*
dworja@hotmail.com

1° Entrega: 13/09/2011

2° Entrega: 27/09/2011

3° Entrega: 25/10/2011

2do. Cuatrimestre de 2011

66.20 Organización de Computadoras

Facultad de Ingeniería, Universidad de Buenos Aires

Resumen

En el presente trabajo práctico se describirán todos los pasos y conclusiones relacionadas al desarrollo e implementación de algunas funcionalidades del comando Tac de Unix.

Índice

1. Introducción	3
2. Implementación	3
2.1. Lenguaje	3
2.2. Descripción del programa	3
2.2.1. Funciones del programa	4
2.2.2. Errores posibles	5
2.3. Desarrollo de actividades	5
2.4. Screenshots	6
3. Conclusiones	14
4. 2º Entrega	14
4.1. Pruebas	14
5. Tercer Entrega	15
5.1. Pruebas	15
5.2. Conclusiones	16

1. Introducción

Tac (tac en el entorno Linux) es un programa que nos permite ver un archivo línea por línea en orden inverso (desde el final hacia el principio). Concatena archivos y los imprime (por salida estándar) en forma inversa. Tac recibe el/los nombre/s de el/los archivos a procesar. Cuando no se le pasa ningún nombre o bien cuando el nombre del archivo es - significa que va a leer de la entrada estándar (stdin).

2. Implementación

2.1. Lenguaje

Como lenguaje de implementación se eligió ANSI C [?] ya que el mismo permite una alta portabilidad entre diferentes plataformas. El desarrollo del programa se realizó usando un editor de texto (gedit,vim, kwrite) y compilando los archivos fuente con GCC que viene en linux (probado en Ubuntu y Fedora 12). También una parte se desarrollo usando el IDE Eclipse con su respectivo plugin para C/C++ y se comprobó que dicho programa compile con el GCC que viene dentro de NETBSD, y como adicional, también se comprobó que el mismo compile y funcione en Windows por medio del MinGW el cual es una implemetación de los compiladores GCC para la plataforma Win32 que permite migrar la capacidad de este compilador en entornos windows. La compilación en Linux y en BSD, MIPS se realizó con la siguiente línea de comando:

```
gcc -Wall -ansi -O0 -o tp0 Definitions.c Message.c Process_File.c  
Process_Stdin.c Stack.c tp0.c
```

2.2. Descripción del programa

Cuando se pasa un nombre como argumento, se verifica que dicho nombre que está haciendo referencia a un archivo, verdaderamente haga referencia a un archivo (.txt, .dat etc (Regular File)) y no a un archivo de directorio, un archivo de dispositivo etc. Una vez hecha la verificación el programa se dispone a leer cada una de las líneas desde el final hacia el principio y las vuelca por salida estándar (stdout). En caso que las lineas provengan del stdin, se utiliza un buffer de memoria dinámica que almacena todas las lineas.

La función `main` se encuentra en `tp0.c` y se encarga de interpretar las opciones y argumentos. En caso de ser una opción, como ayuda o versión, se imprime el mensaje correspondiente y finaliza la ejecución. Cuando no es una opción de ayuda o versión, se procede a procesar los datos de entrada, ya sean provenientes del stdin o de archivos. De acuerdo a cuál sea el caso, se utilizan las funciones `Process_Stdin` ó `Process_File` . La salida de estas funciones proveen un código de error que sirve como salida del programa. Los mensajes de versión y ayuda se imprimen por salida estándar (stdout) y el programa finaliza devolviendo 0 (cero) al sistema. Los mensajes de error se imprimen por la salida de errores (stderr) y el programa finaliza devolviendo 1 (uno) al sistema.

2.2.1. Funciones del programa

El programa cuenta con una serie de funciones las cuales hacen al funcionamiento principal del mismo, como a otros aspectos relacionados a brindar facilidad de uso al usuario dándole toda la información necesaria para su correcto uso.

int Process_File(char* filename,int argc,int idx): Procesa el archivo cuyo nombre es pasado en filename. Los valores de retorno en caso de error pueden ser los siguientes: Devuelve -1 si el archivo es inexistente o cualquier fallo de la función stat(). Devuelve 0 si no es un archivo regular.

int Process_Stdin(): Procesa la entrada estándar. Devuelve 1 si el procesamiento es satisfactorio. En caso contrario devuelve un número distinto de 1.

bool Is_Stdin(int idx,int argc,char* filename): Devuelve true si se va a procesar la entrada estándar o false en caso de que se trate de un archivo

bool Is_Option(char* opt): Devuelve true si el puntero a caracter apunta a alguna de las opciones (-help -h, -V -Version), sino false.

void Print_Option_Msg(char* opt): Imprime el mensaje de ayuda o el mensaje de versión según corresponda, de acuerdo al valor al que hace referencia opt.

int ERROR_MSJ(int code): Imprime un mensaje de error por stderr. El mensaje impreso depende del valor de code. El valor de retorno es 1.

int is_regular(const char* filename): Determina si el nombre al que apunta filename es un archivo regular o no. El valor de retorno puede ser -1 si no existe el archivo (o directorio), un valor distinto de 0 (cero) si es un archivo regular o 0 si no lo es.

void create(Type_Stack* ptr_Stack): Crea una pila vacía

bool empty(Type_Stack* ptr_Stack): Devuelve true si la pila a la que apunta ptr_Stack está vacía, sino false.

int push(Type_Stack* ptr_Stack,char* line): Inserta el puntero a caracter al tope de la pila

void pop(Type_Stack* ptr_Stack): Elimina el tope de la pila

char* top(Type_Stack* ptr_Stack): Devuelve el contenido del tope de la pila

void destroy(Type_Stack* ptr_Stack): Destruye la pila.

int main(int argc, char* argv[]): Es la entrada principal al programa y el que contiene toda la lógica necesaria para procesar los archivos y/o entrada estándar

2.2.2. Errores posibles

1. El procesamiento de la entrada estándar causó el agotamiento del heap.
2. La invocación del programa es incorrecta.
3. Alguno de los archivos no es un archivo regular.
4. Alguno de los archivos es inexistente.

Se contemplan otros errores gracias al uso de la variable externa `errno`. Cuando ocurre un error inesperado, el mismo es informado por `stderr` (con la función `perror()`).

2.3. Desarrollo de actividades

1. Se instaló en un linux un repositorio de fuentes (SVN) para que al dividir las tareas del TP se pudiese hacer una unión de los cambios ingresados por cada uno de los integrantes más fácilmente.
2. Cada persona del grupo se comprometió a que sus cambios en el código fuente y los cambios obtenidos del SVN que pudiesen haber subido los otros integrantes del grupo, sean compilados en diferentes sistemas operativos: Windows, Linux y el GXEmul, asegurando así la máxima portabilidad entre plataformas planteada en el enunciado.
3. Se estableció que todos los integrantes en mayor o menor medida, contribuyan en el desarrollo de todas las partes del código para que nadie quede en desconocimiento de lo que se hizo en cada sección. Si bien cada parte del código está comenazada por diferentes integrantes (parseo de los argumentos, lectura de los ficheros, etc), todos nos familiarizamos con cada una de estas partes y cumplimos la función de testers de lo hecho por otros integrantes.
4. Se propuso como meta paralela, hacer el programa lo mas reutilizable posible tratando de que los métodos desarrollados, sean los suficientemente modulares como para su posible reutilización en los TPs venideros.
5. Debido desconocimiento de \LaTeX [?] para algunos integrantes del grupo, uno de los integrantes dió una breve introducción de como desarrollar este informe para que todos pudiesen modificarlo y agregar lo que considerase necesario. Para compilar el archivo del informe se usaron el compilador de \LaTeX [?] propio que viene en el Linux, y en el caso que sea necesario compilar el informe en un Windows se uso el \LaTeX el cuál es una implementación del compilador de \LaTeX [?] para dicha plataforma.
6. Para poder generar el código assembler a partir del código fuente, dentro de NETBSD se utilizó gcc con la siguiente opción:

```
gcc -S -Wall -O0 Definitions.c -mrnames
gcc -S -Wall -O0 Message.c -mrnames
gcc -S -Wall -O0 Process_File.c -mrnames
gcc -S -Wall -O0 Process_Stdin.c -mrnames
gcc -S -Wall -O0 Stack.c -mrnames
gcc -S -Wall -O0 tp0.c -mrnames
```

7. Para crear el presente informe en formato PDF usando L^AT_EX [?] en Linux, ingresar los siguientes comandos:

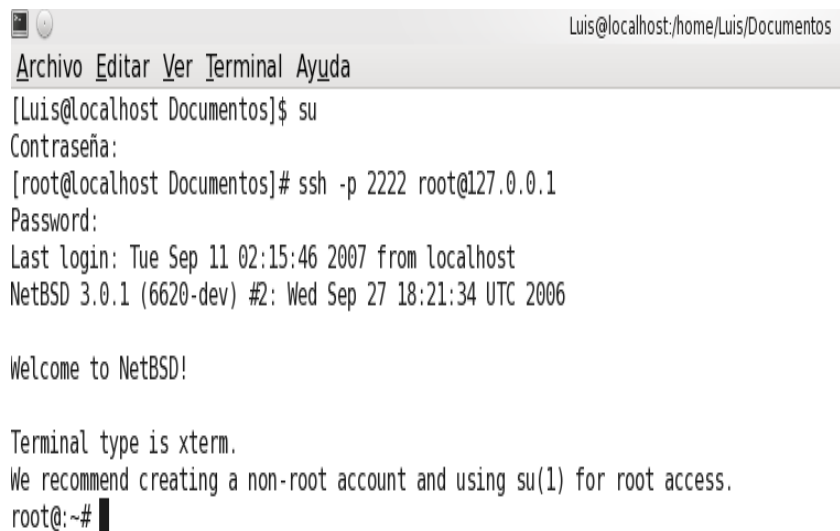
```
$ latex tp0.tex
$ bibtex tp0.aux
$ latex tp0.tex
$ latex tp0.tex
$ dvipdfm tp0.dvi
```

8. Para observar el documento sin necesidad de tener que crear el PDF con dvipdfm, ingresar:

```
$ xdvi tp0.dvi
```

2.4. Screenshots

1. NetBSD corriendo



```
Luis@localhost:/home/Luis/Documentos
Archivo Editar Ver Terminal Ayuda
[Luis@localhost Documentos]$ su
Contraseña:
[root@localhost Documentos]# ssh -p 2222 root@127.0.0.1
Password:
Last login: Tue Sep 11 02:15:46 2007 from localhost
NetBSD 3.0.1 (6620-dev) #2: Wed Sep 27 18:21:34 UTC 2006

Welcome to NetBSD!

Terminal type is xterm.
We recommend creating a non-root account and using su(1) for root access.
root@:~#
```

2. Túnel SSH
3. Linux a NetBSD

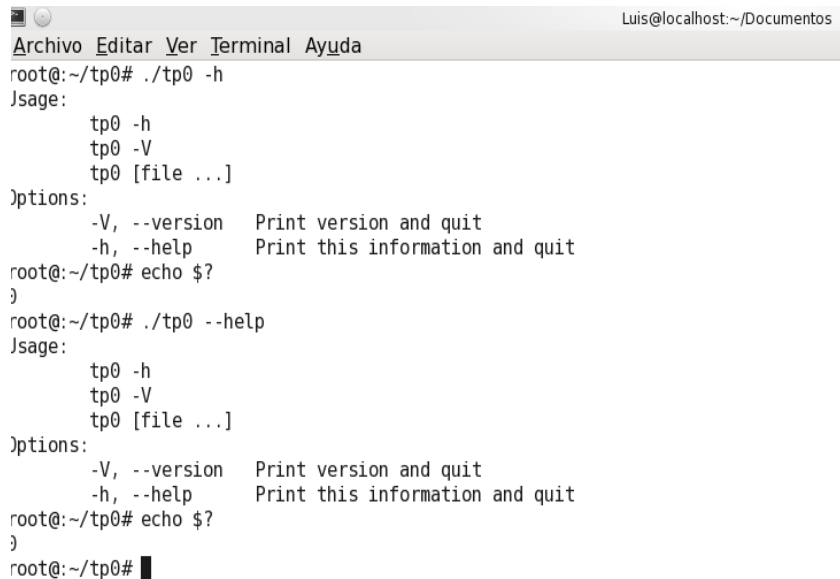
The image shows two terminal windows. The top window is on a Linux host (Luis@localhost) and shows an SCP command being executed to transfer files to a NetBSD host (root@127.0.0.1). The command is: `scp -P2222 -r . root@127.0.0.1:/root/tp0`. The output shows a progress bar for each file being transferred, including `Process_File.c`, `1.txt`, `alumnos.txt`, `Message.c`, `biblia.txt`, `4.txt`, `vacio.txt`, `3.txt`, `Definitions.h`, `2.txt`, `Process_File.h`, `Message.h`, `Stack.c`, `2lineasLargas.txt`, `Process_Stdin.c`, `Process_Stdin.h`, `Stack.h`, `Definitions.c`, `tp0.c`, and `tp0`. The bottom window is on the NetBSD host (root@tp0) and shows the files being received, including `.bash_history`, `.klogin`, `.shrc`, `.viminfo`, `.bashrc`, `.login`, `.ssh`, `.vimrc`, `.cshrc`, `.profile`, `.svn`, and the transferred files: `1.txt`, `2.txt`, `2lineasLargas.txt`, `3.txt`, `4.txt`, `Definitions.c`, `Definitions.h`, `Message.c`, `Message.h`, `Process_File.c`, `Process_File.h`, `Process_Stdin.c`, `Process_Stdin.h`, `Stack.c`, `Stack.h`, `alumnos.txt`, `biblia.txt`, `tp0.c`, and `vacio.txt`.

4. NetBsd a Linux

The image shows a terminal window on a NetBSD host (Luis@localhost/home/Luis/Documentos/TpOrga/Tp0). The command being executed is: `scp -p 2222 Process_File.c Luis@172.20.0.1:/home/Luis/Documentos/TpOrga/Archivos`. The output shows the progress of the transfer: `100% 1923 1.9KB/s 00:00`. The command is successful, and the file `Process_File.c` is transferred to the Linux host.

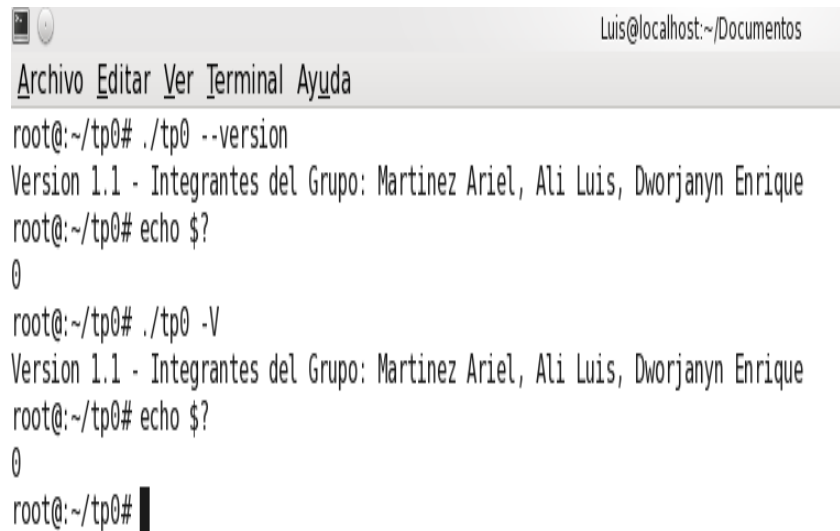
5. Programa compilado y corriendo en NetBSD

6. Mensaje de ayuda



```
Luis@localhost:~/Documentos
Archivo Editar Ver Terminal Ayuda
root@:/tp0# ./tp0 -h
Jsage:
    tp0 -h
    tp0 -V
    tp0 [file ...]
Options:
    -V, --version    Print version and quit
    -h, --help       Print this information and quit
root@:/tp0# echo $?
0
root@:/tp0# ./tp0 --help
Jsage:
    tp0 -h
    tp0 -V
    tp0 [file ...]
Options:
    -V, --version    Print version and quit
    -h, --help       Print this information and quit
root@:/tp0# echo $?
0
root@:/tp0#
```

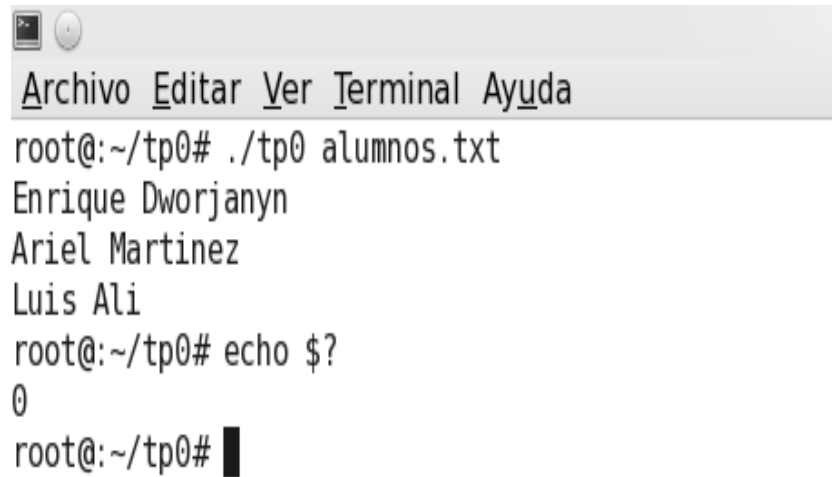
7. Mensaje de Versión



```
Luis@localhost:~/Documentos
Archivo Editar Ver Terminal Ayuda
root@:/tp0# ./tp0 --version
Version 1.1 - Integrantes del Grupo: Martinez Ariel, Ali Luis, Dworjanyn Enrique
root@:/tp0# echo $?
0
root@:/tp0# ./tp0 -V
Version 1.1 - Integrantes del Grupo: Martinez Ariel, Ali Luis, Dworjanyn Enrique
root@:/tp0# echo $?
0
root@:/tp0#
```

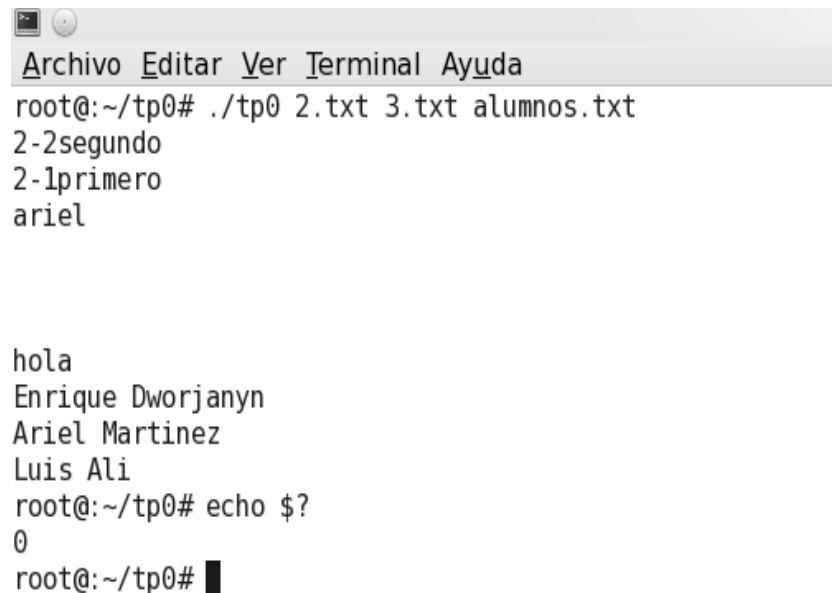

8. Pruebas

9. Archivo Alumnos.txt

A terminal window with a title bar containing icons for file operations and a menu bar with options: Archivo, Editar, Ver, Terminal, Ayuda. The terminal shows the execution of a script that reads from a file named 'alumnos.txt'.

```
root@:~/tp0# ./tp0 alumnos.txt
Enrique Dworjanyn
Ariel Martinez
Luis Ali
root@:~/tp0# echo $?
0
root@:~/tp0#
```

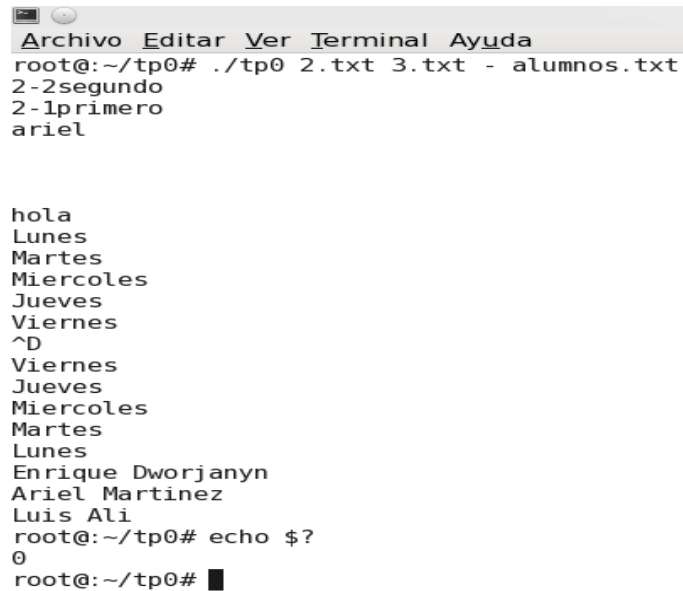
10. Varios Archivos

A terminal window with a title bar containing icons for file operations and a menu bar with options: Archivo, Editar, Ver, Terminal, Ayuda. The terminal shows the execution of a script that reads from multiple files. The output is split into two sections.

```
root@:~/tp0# ./tp0 2.txt 3.txt alumnos.txt
2-2segundo
2-1primero
ariel

hola
Enrique Dworjanyn
Ariel Martinez
Luis Ali
root@:~/tp0# echo $?
0
root@:~/tp0#
```

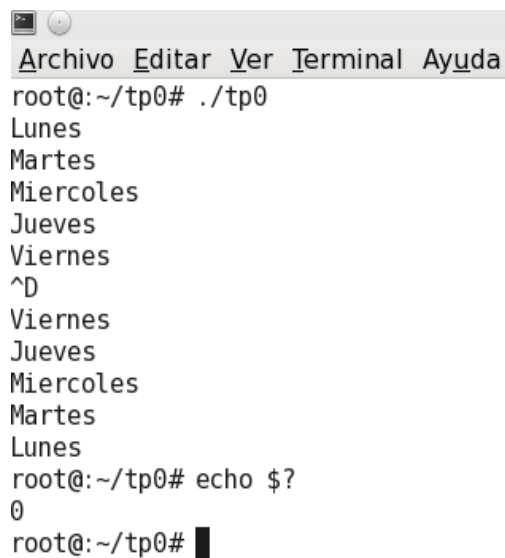
11. Stdin y Archivos



```
Archivo Editar Ver Terminal Ayuda
root@:/tp0# ./tp0 2.txt 3.txt - alumnos.txt
2-2segundo
2-1primero
ariel

hola
Lunes
Martes
Miercoles
Jueves
Viernes
^D
Viernes
Jueves
Miercoles
Martes
Lunes
Enrique Dworjanyn
Ariel Martinez
Luis Ali
root@:/tp0# echo $?
0
root@:/tp0#
```

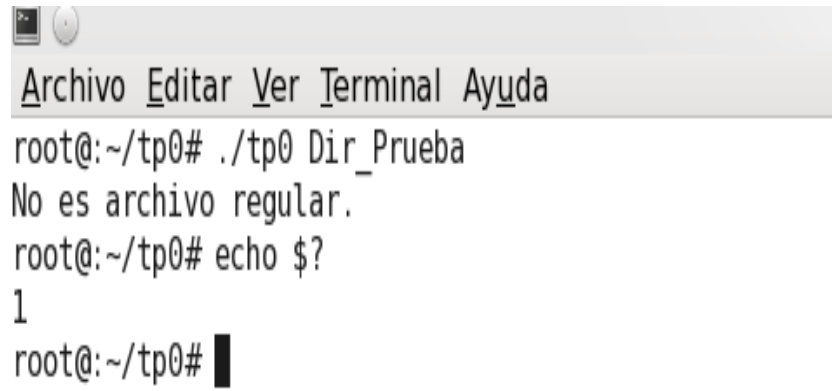
12. Solo Stdin



```
Archivo Editar Ver Terminal Ayuda
root@:/tp0# ./tp0
Lunes
Martes
Miercoles
Jueves
Viernes
^D
Viernes
Jueves
Miercoles
Martes
Lunes
root@:/tp0# echo $?
0
root@:/tp0#
```

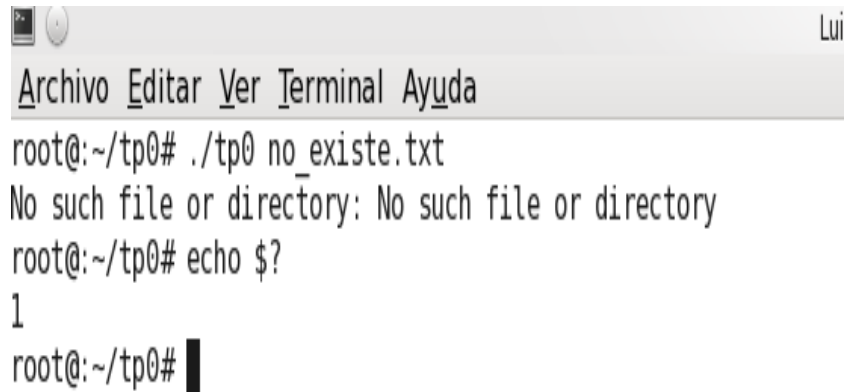
13. Errores

14. Archivo regular.

A terminal window with a title bar containing icons and the text "Archivo Editar Ver Terminal Ayuda". The terminal shows a root user at a prompt in the directory ~/tp0. The user enters the command ./tp0 Dir_Prueba, which results in an error message "No es archivo regular.". The user then enters the command echo \$?, which outputs the number 1. The prompt returns to root@:~/tp0# with a cursor.

```
Archivo Editar Ver Terminal Ayuda
root@:~/tp0# ./tp0 Dir_Prueba
No es archivo regular.
root@:~/tp0# echo $?
1
root@:~/tp0#
```

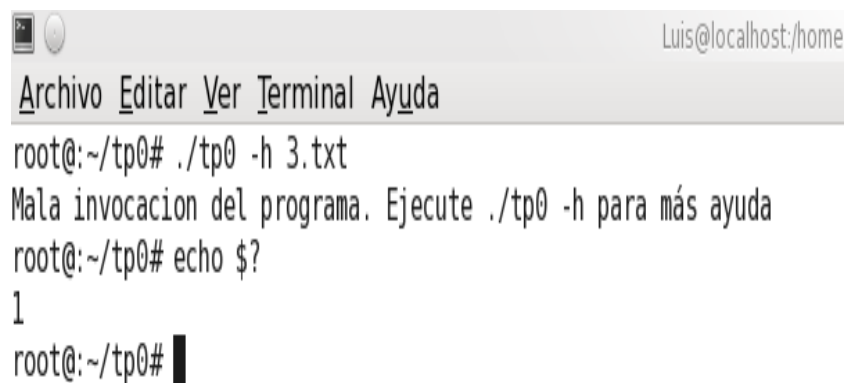
15. Archivo inexistente.



A terminal window titled "Lui" with a menu bar containing "Archivo", "Editar", "Ver", "Terminal", and "Ayuda". The prompt is "root@:~/tp0#". The user enters the command `./tp0 no_existe.txt`, and the terminal outputs "No such file or directory: No such file or directory". The user then enters `echo $?`, and the terminal outputs "1". The prompt is now "root@:~/tp0#".

```
root@:~/tp0# ./tp0 no_existe.txt
No such file or directory: No such file or directory
root@:~/tp0# echo $?
1
root@:~/tp0#
```

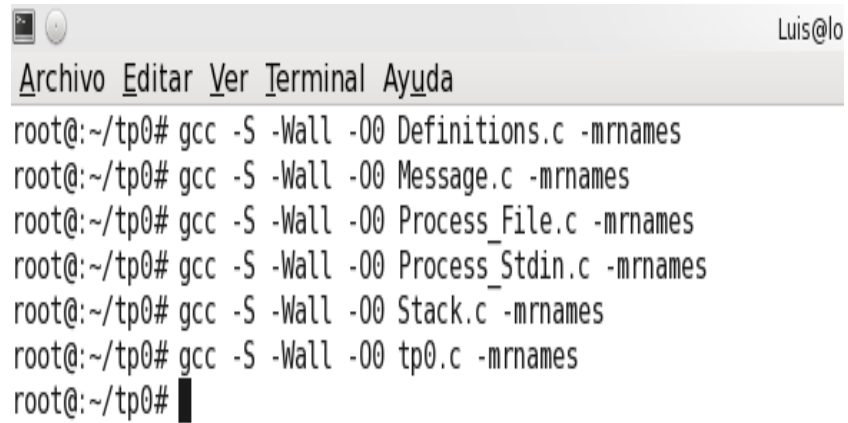
16. Mala Invocación.



A terminal window titled "Luis@localhost:/home" with a menu bar containing "Archivo", "Editar", "Ver", "Terminal", and "Ayuda". The prompt is "root@:~/tp0#". The user enters the command `./tp0 -h 3.txt`, and the terminal outputs "Mala invocacion del programa. Ejecute ./tp0 -h para más ayuda". The user then enters `echo $?`, and the terminal outputs "1". The prompt is now "root@:~/tp0#".

```
root@:~/tp0# ./tp0 -h 3.txt
Mala invocacion del programa. Ejecute ./tp0 -h para más ayuda
root@:~/tp0# echo $?
1
root@:~/tp0#
```

17. Código assembler



A terminal window with a title bar containing icons for file, edit, view, and help, and the text "Luis@lo". The menu bar shows "Archivo", "Editar", "Ver", "Terminal", and "Ayuda". The terminal content shows a series of gcc compilation commands in the directory ~/tp0, each with the -Wall and -mrnames flags. The commands are: gcc -S -Wall -O0 Definitions.c -mrnames, gcc -S -Wall -O0 Message.c -mrnames, gcc -S -Wall -O0 Process_File.c -mrnames, gcc -S -Wall -O0 Process_Stdin.c -mrnames, gcc -S -Wall -O0 Stack.c -mrnames, gcc -S -Wall -O0 tp0.c -mrnames, and a final prompt root@:~/tp0# with a cursor.

```
root@:~/tp0# gcc -S -Wall -O0 Definitions.c -mrnames
root@:~/tp0# gcc -S -Wall -O0 Message.c -mrnames
root@:~/tp0# gcc -S -Wall -O0 Process_File.c -mrnames
root@:~/tp0# gcc -S -Wall -O0 Process_Stdin.c -mrnames
root@:~/tp0# gcc -S -Wall -O0 Stack.c -mrnames
root@:~/tp0# gcc -S -Wall -O0 tp0.c -mrnames
root@:~/tp0# █
```

3. Conclusiones

1. Si bien el lo solicitado por el programa no era excesivamente difícil, la realización completa del TP llevó cierta dificultad al tener que realizarlo en el contexto solicitado: alta portabilidad, desarrollo en C, e informe hecho en \LaTeX [?].
2. En el primer caso la dificultad radicaba en tener configurado y funcionando el GXEmul dentro de un Linux, y lograr que en ambos casos el programa compile y corra sin problemas.
3. Debido a nuestro desconocimiento con \LaTeX [?], tuvimos que invertir tiempo en encontrar forma de realizar el presente documento de la manera más correcta posible
4. En cuanto al trabajo grupal en si mismo, no hubo inconvenientes de ningún tipo ya que al ser el grupo relativamente chico y tener conocimiento del manejo del versionado de un proyecto ante cambios ingresado por los integrantes (por medio del SVN), la introducción de modificaciones y correcciones fué fluida.

4. 2º Entrega

4.1. Pruebas

El conjunto de pruebas se escribieron en un script de linux adjunto en la siguiente sección. La mayoría de las pruebas se basan en comparar la salida de nuestro programa con el comando tac. Para el caso de NetBSD, al no encontrarse el comando tac se optó por hacer una doble ejecución del tp y comparar la salida con el archivo original. A continuación se comentan cada uno:

Opción de ayuda:

\$tp0 -h

Usage:

tp0 -h

tp0 -V

tp0 [file ...]

Options:

-V, --version Print version and quit

-h, --help Print this information and quit

Opción de versión:

\$tp0 -V

Version 1.1 - Integrantes del Grupo: Martinez, Ariel, Ali, Luis, Dworjanyyn Enrique

Un archivo vacío:

\$tp0 vacio.txt

Ejemplo básico con un archivo de 2 lineas:

\$/tp0 2Lineas.txt

Archivo con lineas y saltos:

\$/tp0 lineasYsaltos.txt

Archivo con 2 lineas excesivamente largas:

\$tp0 2lineasLargas.txt

Archivo inexistente:

\$tp0 archivoInexistente.txt

Recibe dos archivos como parámetros:

\$/tp0 2Lineas.txt lineasYsaltos.txt

Un conjunto de 10 pruebas de archivos binarios aleatorios utilizando /etc/dev/urandom

5. Tercer Entrega

5.1. Pruebas

Se modificó el script de pruebas. En este caso existen 2 script, cuyo funcionamiento se detalla a continuación:

generar_pruebas_y_res.sh

Este script en primera instancia verifica la existencia de ciertos directorios para poder volcar el resultado. Si no existen, los crea. Estos directorios son:

Casos_de_prueba	Salida	Salida_Esperada
Casos_de_prueba/binario	Salida/Binario	Salida_Esperada/Binario
Casos_de_prueba/Texto	Salida/Texto	Salida_Esperada/Texto

Una vez creado los directorios, el script genera archivos binarios en la carpeta

Casos_de_prueba/binario. Estos archivos son de distintos tamaños que van desde 0 kb a 10 kb. El usuario se debe encargar de poner a mano los archivos que el mismo genera para hacer pruebas, en la carpeta Casos_de_prueba/Texto. A Continuación el script ejecuta el comando tac para los archivos que estan en Casos_de_prueba/binario redirigiendo la salida del tac a un archivo con el mismo nombre que el archivo de entrada del tac pero agregando .ref al final. Este archivo queda guardado en la carpeta Salida_Esperada/Binario.

De la misma forma, el script ejecuta el comando tac sobre los archivos de la carpeta

Casos_de_prueba/Texto y vuelca los archivos (resultado de aplicar tac a cada

uno), en la carpeta Salida_Esperada/Texto.

Una vez hecho esto, se debe copiar a NetBSD por medio del tunel SSH, las carpetas (con todo su contenido) Casos_de_prueba, Salida y Salida_Esperada, y el archivo test.sh

\$/test.sh

Este script se encarga de ejecutar el tp pasando a los archivos de la carpeta Casos_de_prueba/binario y Casos_de_prueba/Texto tanto por stdin como por argumento.

Lo que imprime el tp por stdout se redirige a un archivo a la carpeta Salida/Binario o Salida/Texto, según corresponda.

Una vez hecho esto el script simplemente compara (usando el comando diff) al archivo que esta en Salida/binario (o /Texto) con el que corresponde a la salida del tac que se encuentra en Salida_Esperada/Binario (o /Texto).

Se van listando los archivos a medida que la salida del tp para los mismos coincide con la salida del tac. Si alguno tiene diferencia, se indica que el archivo falló y el script finaliza su ejecución.

Cabe aclarar que para que este script funcione, el nombre del ejecutable que se crea al compilar los fuentes del tp se debe llamar ***tp0***

5.2. Conclusiones

Se corrigieron solamente 2 errores en el código (2 errores con respecto a la primer entrega) para que todas las pruebas sean satisfactorias.

Cuando se procesaba la entrada estándar, se estaba agregando un \0 al final de un puntero a char, dado que la función printf imprime hasta encontrar un \0 .

Al no poner \0, imprimía la línea con

un resto de basura y la entrada no coincidía. (Este error estaba en el archivo Process_Stdin en el método sowh_all_lines.) Cuando se procesaba un archivo cuyo nombre se indicaba por argumento,

se estaba usando como fin de línea tanto \n como \0.

Usando solo \n pasaron todas las pruebas.