Versión en C del comando rev

Lucas Simonelli, *Padrón Nro. 93111*
lucasp.simonelli@gmail.com

Tomás Boccardo, *Padrón Nro. 00.000*
dirección de e-mail

Andrés Sanabria, *Padrón Nro. 00.000*
dirección de e-mail

2do. Cuatrimestre de 2013
66.20 Organización de Computadoras – Práctica Martes
Facultad de Ingeniería, Universidad de Buenos Aires

1. Introducción

En el presente trabajo práctico se elaboró una versión en C del comando *rev* de Unix[1], que puede ejecutarse sin problemas tanto en linux sobre x86 como en netBSD sobre MIPS32.

2. Función

El ejecutable tendrá el mismo objeto que el comando rev, es decir, leerá un archivo por alguno de los canales ofrecidos y lo invertirá línea a línea.

3. Comandos para la compilacion

Para compilar el programa, deberá introducirse el siguiente comando:

```
#compilamos
~$gcc -o tp0 main.c

#corremos
~$./tp0 [archivo]
```

3.1. Sintaxis de uso

```
$ ./tp0 -h
Usage:
./tp0 -h
./tp0 -V
./tp0 [file...]
Options:
-V, --version, print version and quit.
-h, --help, print this information and quit.

Examples:
./tp0 foo.txt bar.txt
./tp0 gz.txt
echo "Hola mundo" | ./tp0
```

4. Corridas de prueba

(Los archivos se facilitan junto con este informe en la entrega digital)

4.1. Corrida con archivo de parámetro

```
~$./tp0 ejemplo.txt
1elif ed aenil aremirp al se atsE
.adnuges al se atse y
```

[1] http://linux.about.com/library/cmd/blcmdl_rev.htm

4.2. Corrida con entrada de stdin

```
~$echo "Hola mundo" | ./tp0
odnum aloH
```

5. Código fuente

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>

#define TAM_INI_CADENA 40


//Lee una linea de tamanio arbitrario. Devuelve NULL al llegar a EOF
char* leerLinea(FILE* archivo){
    int tam = TAM_INI_CADENA, i=0;
    char *linea = (char*)malloc(sizeof(char)*tam);
    char letra;
    do {
        letra = fgetc(archivo);
        linea[i]=letra;
        if (tam==i+1){
            tam+=10;
            char *aux=(char*) realloc(linea,sizeof(char)*tam);
            if (!aux) {
                linea[i]='\0';
                return linea;
            } else {
                linea=aux;
            }
        }
        i++;
    } while (letra!='\n' && letra!=EOF);
    linea[i-1]='\0';

        if (i-1 == 0 ){
                free(linea);
                return NULL;
        }
    return linea;
}


void invertirLinea(char* linea){
    if (!linea)
        return;
    int len = strlen(linea);
```

```
    int i = 0;
    int l = len-1;
    while (l > i){
        //Swap
        char aux = linea[i];
        linea[i]=linea[l];
        linea[l]=aux;
        i++;
        l--;
    }

}


int main(int argc, char** argv){
        //FILE* ejemplo = fopen("ejemplo","r");
        int nFiles = argc - 1;
        FILE* file;
        bool noFile = false;


        if (nFiles == 0){
                file = stdin;
                nFiles = 1;
                noFile = true;
        }

        int i = 0;

        while (i < nFiles){
                if (! noFile){
                        file = fopen(argv[i+1],"r");
                }

                while (!feof(file)){
                        char* s=leerLinea(file);
                        invertirLinea(s);
                        if (s){
                                printf("%s\n",s);
                                free(s);
                        }
                }

                i++;
                if (! noFile){
                        fclose(file);
                }
        }

        return 0;
```

}

6. Código MIPS32

7. Conclusiones

Se aprendieron las generalidades del uso de gxemule.

8. Enunciado

66:20 Organización de Computadoras
Trabajo práctico 0: Infraestructura básica

1. Objetivos

Familiarizarse con las herramientas de software que usaremos en los siguientes trabajos, implementando un programa (y su correspondiente documentación) que resuelva el problema piloto que presentaremos más abajo.

2. Alcance

Este trabajo práctico es de elaboración grupal, evaluación individual, y de carácter obligatorio para todos alumnos del curso.

3. Requisitos

El trabajo deberá ser entregado personalmente, en la fecha estipulada, con una carátula que contenga los datos completos de todos los integrantes.

Además, es necesario que el trabajo práctico incluya (entre otras cosas, ver sección 6), la presentación de los resultados obtenidos, explicando, cuando corresponda, con fundamentos reales, las causas o razones de cada resultado obtenido.

El informe deberá respetar el modelo de referencia que se encuentra en el grupo[1], y se valorarán aquellos escritos usando la herramienta TEX / LATEX.

4. Recursos

Usaremos el programa GXemul [2] para simular el entorno de desarrollo que utilizaremos en este y otros trabajos prácticos, una máquina MIPS corriendo una versión reciente del sistema operativo NetBSD [3].

En la clase del 20/8 hemos repasado, brevemente, los pasos necesarios para la instalación y configuración del entorno de desarrollo.

[1] `http://groups.yahoo.com/group/orga-comp`

5. Implementación

5.1. Programa

El programa, a escribir en lenguaje C, es una versión del comando `rev` de UNIX. El mismo concatena y escribe en `stdout` el contenido de uno o mas archivos, invirtiendo el orden de los caracteres de cada línea. En nuestro caso, se asume que cada caracter mide 1 byte.

Retorna 0 sólo en caso de éxito. Todos los errores deben ser impresos por `stderr`.

5.1.1. Parámetros

El programa debe recibir como parámetros los archivos sobre los que va a trabajar. En caso de no recibir ningún parámetro, entonces deberá operar sobre los datos provenientes de `stdin`. Debido a que el tamaño de la información a recibir es desconocida, debe utilizarse memoria dinámica. Cualquier error debe ser informado debidamente.

5.2. Ejemplos

Primero, usamos la opción `-h` para ver el mensaje 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.
Examples:
   tp0 foo.txt bar.txt
   tp0 gz.txt
```

A continuación, ejecutamos algunas pruebas:

```
$ cat > file1
Esta es la primera línea de file1
 y esta es la segunda.

$ ./tp0 file1
1elif ed aeníl aremirp al se atsE
.adnuges al se atse y

$ cat > file2
Ahora es el turno
de file2.

$ ./tp0 return.txt status.txt
1elif ed aeníl aremirp al se atsE
.adnuges al se atse y
onrut le se arohA
.2elif ed

$ cat file1 | ./tp0
1elif ed aeníl aremirp al se atsE
.adnuges al se atse y

$ ./tp0 file1 | rev
Esta es la primera línea de file1
 y esta es la segunda.
```

5.3. Portabilidad

Como es usual, es necesario que la implementación desarrollada provea un grado mínimo de portabilidad. Para satisfacer esto, el programa deberá funcionar al menos en NetBSD/pmax (usando el simulador GXemul [2]) y la versión de Linux (Knoppix, RedHat, Debian, Ubuntu) usada para correr el simulador, Linux/i386.

6. Informe

El informe deberá incluir:

- Documentación relevante al diseño e implementación del programa;
- Comando(s) para compilar el programa;
- Las corridas de prueba, con los comentarios pertinentes;
- El código fuente, en lenguaje C;
- El código MIPS32 generado por el compilador;
- Este enunciado.

7. Fechas

- Entrega: 10/9/2013.
- Última revisión: 17/9/2013.
- Vencimiento: 24/9/2013.

Referencias

[1] NetBSD rev manual page. `http://netbsd.gw.com/cgi-bin/man-cgi?rev+1.NONE+NetBSD-current`.

[2] GXemul, `http://gavare.se/gxemul/`.

[3] The NetBSD project, `http://www.netbsd.org/`.