

Versión en C del comando rev

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

Tomás Boccardo, *Padrón Nro. 93637*
tomasboccardo@gmail.com

Andrés Sanabria, *Padrón Nro. 93403*
andresg.sanabria@gmail.com

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

Resumen

El presente trabajo consiste en la implementación de un programa, similar al comando rev de Unix, encargado de concatenar y escribir en stdout el contenido invertido de cada línea, de uno o más archivos pasados como parámetros. El mismo contiene adjunto el código fuente en lenguaje C, el código MIPS32 generado por el compilador y un instructivo para la compilación de código fuente. Además se incluyen algunos ejemplos de uso del programa, con el objetivo de mostrar su funcionalidad completamente, y se mencionan ciertos aspectos que consideramos importantes sobre el diseño e implementación del Programa.

1. Introducción

El objetivo del trabajo desarrollado en este informe es familiarizarse con las herramientas a utilizar en trabajos posteriores. Entre ellas se destacan el compilador GCC y el programa GXemul, el cual se utilizará para simular un entorno de desarrollo de una máquina MIPS corriendo una versión del sistema operativo NetBSD. Para ello se implementará un programa con funcionalidad similar al rev de Unix¹.

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. Desarrollo

3.1. Diseño e Implementación

Algunas suposiciones realizadas fueron las siguientes:

- Cada caracter mide 1 byte.
- El programa recibe como parámetros los archivos sobre los que trabajar. En caso de no recibir ningún parámetro, opera sobre los datos provenientes de stdin.
- Dividimos nuestro programa en 3 módulos básicos, cada uno de los cuales tiene una tarea específica.

leerLinea: Se encarga, como su nombre lo indica, de leer caracter por caracter el contenido de una línea hasta encontrar el fin de archivo o el caracter de fin de línea. Recibe como parámetro el descriptor del archivo del que debe leer o el descriptor de stdin si este fuera el caso. Devuelve un buffer con el contenido de la línea.

invertirLinea: Este módulo se utiliza para invertir el orden de aparición de los caracteres en una línea. De este modo, el último caracter de la línea quedará en primer lugar y el primero último, y de forma análoga se intercambiarán el resto de los caracteres. Recibe como parámetro un buffer con la línea en su estado original y devuelve en ese mismo buffer, la línea invertida.

main: Por último, tenemos la función principal que se encarga de interpretar los parámetros con los que fue llamado el programa y llamar a los módulos antes mencionados cuando sea necesario. De acuerdo a los parámetros que le sean ingresados el programa se comportará de distinta manera:

Si recibe como parámetro '-h' muestra por pantalla una breve explicación del uso del programa.

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

En caso de recibir el parámetro '-V' muestra la versión del programa en ejecución.

De no recibir parámetros lee por entrada standard e invierte el contenido de cada línea.

En otro caso, abre e invierte las líneas de los archivos pasados como parámetro. En caso de no encontrar alguno de ellos, informa por stdout el nombre del archivo que no encontró.

4. Comandos para la compilacion

Para compilar el programa, deberá introducirse el siguiente comando:

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

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

4.1. 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
```

5. Corridas de prueba

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

5.1. Corrida con archivo de parámetro

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

5.2. Corrida con entrada de stdin

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

6. Código fuente

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

#define TAMINLCADENA 40

//Lee una linea de tamaño arbitrario. Devuelve NULL al llegar a EOF
char* leerLinea(FILE* archivo){
    int tam = TAMINLCADENA, 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;
    }
    else if (strcmp(argv[1],"-h")==0 && (nFiles==1)){
        printf("Usage:\ntp0 -h\ntp0 -V\ntp0 [ file ... ] \nOptions:\n-V, -"
        return 0;
    }
    else if (strcmp(argv[1],"-V")==0 && (nFiles==1)){
        printf("Tp0 Version 1.0");
        return 0;
    }

    int i = 0;

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

        if (! file){
            fprintf(stderr,"File %s not found\n",argv[i+1]);
        }
        else{
            while (!feof(file)){
                char* s=leerLinea(file);
                invertirLinea(s);
                if (s){
                    printf("%s\n",s);
                    free(s);
                }
            }
        }
    }
}

```

```

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

    return 0;
}

```

7. Código MIPS32

```

.file    1 "poder.c"
.section .mdebug.abi32
.previous
.abicalls
.text
.align   2
.globl   leerLinea
.ent     leerLinea
leerLinea:
    .frame    $fp,64,$ra                # vars= 24, regs= 3/0, args= 16, extra=
    .mask     0xd0000000,-8
    .fmask    0x00000000,0
    .set      noreorder
    .cpload   $t9
    .set      reorder
    subu      $sp,$sp,64
    .cpstore  16
    sw        $ra,56($sp)
    sw        $fp,52($sp)
    sw        $gp,48($sp)
    move      $fp,$sp
    sw        $a0,64($fp)
    li        $v0,40                    # 0x28
    sw        $v0,24($fp)
    sw        $zero,28($fp)
    lw        $a0,24($fp)
    la        $t9, malloc
    jal       $ra,$t9
    sw        $v0,32($fp)

$L18:
    lw        $a0,64($fp)
    la        $t9, fgets
    jal       $ra,$t9
    sb        $v0,36($fp)
    lw        $v1,32($fp)
    lw        $v0,28($fp)
    addu      $v1,$v1,$v0
    lbu       $v0,36($fp)
    sb        $v0,0($v1)

```

```

        lw      $v0,28($fp)
        addu    $v1,$v0,1
        lw      $v0,24($fp)
        bne     $v0,$v1,$L21
        lw      $v0,24($fp)
        addu    $v0,$v0,10
        sw      $v0,24($fp)
        lw      $a0,32($fp)
        lw      $a1,24($fp)
        la      $t9,realloc
        jal     $ra,$t9
        sw      $v0,40($fp)
        lw      $v0,40($fp)
        bne     $v0,$zero,$L22
        lw      $v1,32($fp)
        lw      $v0,28($fp)
        addu    $v0,$v1,$v0
        sb      $zero,0($v0)
        lw      $v0,32($fp)
        sw      $v0,44($fp)
        b       $L17
$L22:
        lw      $v0,40($fp)
        sw      $v0,32($fp)
$L21:
        lw      $v0,28($fp)
        addu    $v0,$v0,1
        sw      $v0,28($fp)
        lb      $v1,36($fp)
        li      $v0,10                                # 0xa
        beq     $v1,$v0,$L19
        lb      $v1,36($fp)
        li      $v0,-1                                # 0xffffffffffffffff
        bne     $v1,$v0,$L18
$L19:
        lw      $v1,32($fp)
        lw      $v0,28($fp)
        addu    $v0,$v1,$v0
        addu    $v0,$v0,-1
        sb      $zero,0($v0)
        lw      $v1,28($fp)
        li      $v0,1                                # 0x1
        bne     $v1,$v0,$L26
        lw      $a0,32($fp)
        la      $t9,free
        jal     $ra,$t9
        sw      $zero,44($fp)
        b       $L17
$L26:
        lw      $v0,32($fp)

```

```

    sw      $v0,44($fp)
$L17:
    lw      $v0,44($fp)
    move    $sp,$fp
    lw      $ra,56($sp)
    lw      $fp,52($sp)
    addu    $sp,$sp,64
    j       $ra
    .end    leerLinea
    .size   leerLinea,.-leerLinea
    .align  2
    .globl  invertirLinea
    .ent    invertirLinea
invertirLinea:
    .frame  $fp,56,$ra                                # vars= 16, regs= 3/0, args= 16, extra=
    .mask   0xd0000000,-8
    .fmask  0x00000000,0
    .set    noreorder
    .cload  $t9
    .set    reorder
    subu    $sp,$sp,56
    .cprestore 16
    sw      $ra,48($sp)
    sw      $fp,44($sp)
    sw      $gp,40($sp)
    move    $fp,$sp
    sw      $a0,56($fp)
    lw      $v0,56($fp)
    bne     $v0,$zero,$L28
    b       $L27
$L28:
    lw      $a0,56($fp)
    la      $t9,strlen
    jal     $ra,$t9
    sw      $v0,24($fp)
    sw      $zero,28($fp)
    lw      $v0,24($fp)
    addu    $v0,$v0,-1
    sw      $v0,32($fp)
$L29:
    lw      $v0,32($fp)
    lw      $v1,28($fp)
    slt     $v0,$v1,$v0
    bne     $v0,$zero,$L31
    b       $L27
$L31:
    lw      $v1,56($fp)
    lw      $v0,28($fp)
    addu    $v0,$v1,$v0
    lbu     $v0,0($v0)

```



```

        sb      $v0,36($fp)
        lw      $v1,56($fp)
        lw      $v0,28($fp)
        addu    $a0,$v1,$v0
        lw      $v1,56($fp)
        lw      $v0,32($fp)
        addu    $v0,$v1,$v0
        lbu     $v0,0($v0)
        sb      $v0,0($a0)
        lw      $v1,56($fp)
        lw      $v0,32($fp)
        addu    $v1,$v1,$v0
        lbu     $v0,36($fp)
        sb      $v0,0($v1)
        lw      $v0,28($fp)
        addu    $v0,$v0,1
        sw      $v0,28($fp)
        lw      $v0,32($fp)
        addu    $v0,$v0,-1
        sw      $v0,32($fp)
        b       $L29
$L27:
        move    $sp,$fp
        lw      $ra,48($sp)
        lw      $fp,44($sp)
        addu    $sp,$sp,56
        j       $ra
        .end    invertirLinea
        .size   invertirLinea,.-invertirLinea
        .rdata
        .align  2
$LC0:
        .ascii  "-h\000"
        .align  2
$LC1:
        .ascii  " Usage:\n"
        .ascii  " tp0 -h\n"
        .ascii  " tp0 -V\n"
        .ascii  " tp0 [ file ...]\n"
        .ascii  " Options:\n"
        .ascii  "-V, --version  Print version and quit.\n"
        .ascii  "-h, --help   Print this information and quit.\n"
        .ascii  " Examples:\n"
        .ascii  " tp0 foo.txt bar.txt\n"
        .ascii  " tp0 gz.txt\n\000"
        .align  2
$LC2:
        .ascii  "-V\000"
        .align  2
$LC3:

```

```

        .ascii  "Tp0 Version 1.0\000"
        .align  2
$LC4:
        .ascii  "r\000"
        .align  2
$LC5:
        .ascii  "File %s not found\n\000"
        .align  2
$LC6:
        .ascii  "%s\n\000"
        .text
        .align  2
        .globl  main
        .ent    main

main:
        .frame   $fp,64,$ra                # vars= 24, regs= 3/0, args= 16, extra=
        .mask    0xd0000000,-8
        .fmask   0x00000000,0
        .set     noreorder
        .cload   $t9
        .set     reorder
        .subu     $sp,$sp,64
        .cprestore 16
        sw       $ra,56($sp)
        sw       $fp,52($sp)
        sw       $gp,48($sp)
        move     $fp,$sp
        sw       $a0,64($fp)
        sw       $a1,68($fp)
        lw       $v0,64($fp)
        addu     $v0,$v0,-1
        sw       $v0,24($fp)
        sb       $zero,32($fp)
        lw       $v0,24($fp)
        bne      $v0,$zero,$L33
        la       $v0, __sF
        sw       $v0,28($fp)
        li       $v0,1                      # 0x1
        sw       $v0,24($fp)
        li       $v0,1                      # 0x1
        sb       $v0,32($fp)
        b        $L34
$L33:
        lw       $v0,68($fp)
        addu     $v0,$v0,4
        lw       $a0,0($v0)
        la       $a1,$LC0
        la       $t9,strcmp
        jal      $ra,$t9
        bne      $v0,$zero,$L35

```

```

        lw      $v1,24($fp)
        li      $v0,1                                # 0x1
        bne     $v1,$v0,$L35
        la      $a0,$LC1
        la      $t9,printf
        jal     $ra,$t9
        sw      $zero,44($fp)
        b       $L32
$L35:
        lw      $v0,68($fp)
        addu    $v0,$v0,4
        lw      $a0,0($v0)
        la      $a1,$LC2
        la      $t9,strcmp
        jal     $ra,$t9
        bne     $v0,$zero,$L34
        lw      $v1,24($fp)
        li      $v0,1                                # 0x1
        bne     $v1,$v0,$L34
        la      $a0,$LC3
        la      $t9,printf
        jal     $ra,$t9
        sw      $zero,44($fp)
        b       $L32
$L34:
        sw      $zero,36($fp)
$L38:
        lw      $v0,36($fp)
        lw      $v1,24($fp)
        slt     $v0,$v0,$v1
        bne     $v0,$zero,$L40
        b       $L39
$L40:
        lbu     $v0,32($fp)
        bne     $v0,$zero,$L41
        lw      $v0,36($fp)
        sll     $v1,$v0,2
        lw      $v0,68($fp)
        addu    $v0,$v1,$v0
        addu    $v0,$v0,4
        lw      $a0,0($v0)
        la      $a1,$LC4
        la      $t9,fopen
        jal     $ra,$t9
        sw      $v0,28($fp)
$L41:
        lw      $v0,28($fp)
        bne     $v0,$zero,$L44
        lw      $v0,36($fp)
        sll     $v1,$v0,2

```

```

        lw      $v0,68($fp)
        addu    $v0,$v1,$v0
        addu    $v0,$v0,4
        la      $a0,--sF+176
        la      $a1,$LC5
        lw      $a2,0($v0)
        la      $t9,fprintf
        jal     $ra,$t9
        b       $L43
$L44:
        lw      $v0,28($fp)
        lhu     $v0,12($v0)
        srl     $v0,$v0,5
        andi    $v0,$v0,0x1
        beq     $v0,$zero,$L46
        b       $L43
$L46:
        lw      $a0,28($fp)
        la      $t9,leerLinea
        jal     $ra,$t9
        sw      $v0,40($fp)
        lw      $a0,40($fp)
        la      $t9,invertirLinea
        jal     $ra,$t9
        lw      $v0,40($fp)
        beq     $v0,$zero,$L44
        la      $a0,$LC6
        lw      $a1,40($fp)
        la      $t9,printf
        jal     $ra,$t9
        lw      $a0,40($fp)
        la      $t9,free
        jal     $ra,$t9
        b       $L44
$L43:
        lw      $v0,36($fp)
        addu    $v0,$v0,1
        sw      $v0,36($fp)
        lbu     $v0,32($fp)
        bne     $v0,$zero,$L38
        lw      $v0,28($fp)
        beq     $v0,$zero,$L38
        lw      $a0,28($fp)
        la      $t9,fclose
        jal     $ra,$t9
        b       $L38
$L39:
        sw      $zero,44($fp)
$L32:
        lw      $v0,44($fp)

```

```

move    $sp,$fp
lw      $ra,56($sp)
lw      $fp,52($sp)
addu    $sp,$sp,64
j       $ra
.end     main
.size   main,.-main
.ident  "GCC: (GNU) 3.3.3 (NetBSD nb3 20040520)"

```

8. Conclusiones

La utilización del emulador GXemul nos permitió simular una máquina MIPS en la que corrimos como sistema operativo una versión de NETBSD. Además, utilizamos como sistema operativo host Linux en un arquitectura INTEL. Con la compilación de nuestro código C y su posterior ejecución en ambas arquitecturas con sistemas operativos diferentes pudimos comprobar la portabilidad de nuestro código. Por otro lado, luego de utilizar el compilador GCC para obtener el código Assembler de nuestro programa y compararlo con el código C del mismo, pudimos observar la enorme diferencia entre el número de instrucciones de un lenguaje de bajo nivel(Assembler) y uno de alto nivel(C).

9. 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 grupo¹, 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 aenil aremirp al se atseE
.adnuges al se atse y
```

```
$ cat > file2
Ahora es el turno
de file2.
```

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

```
$ cat file1 | ./tp0
1elif ed aenil aremirp al se atseE
.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/>.