Trabajo Práctico 0: Infraestructura básica

Damián Muszalski, *Padrón Nro. 88462*
damianfiuba@gmail.com
Federico García, *Padrón Nro. 93379*
fedeagb@gmail.com
Nicolás Fernandez Lema, *Padrón Nro. 93410*
nicolasfernandezlema@gmail.com

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

10/09/2013

1. Objetivo

El objetivo de este trabajo es familiarizarse con las herramientas que utilizaremos para emular un entorno MIPS32.

2. Introducción

En el presente trabajo práctico, se deben implementar en lenguaje C 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.

3. Implementación

El programa debe leer los datos de entrada desde stdin o bien desde uno o más archivos. La salida del programa debe imprimirse por stdout, mientras que los errores deben imprimirse por stderr. La ayuda y versión del programa puede seleccionarse mediante las opciones -h o -V respectivamente. La implementación del comando debe consiste en una función con el siguiente prototipo: int procesarArchivo(FILE* fd);

4. Desarrollo

A continuación se citan los pasos a que se siguieron para hacer el trabajo práctico.

4.1. Paso 1: Configuración de Entorno de Desarrollo

El primer paso fue configurar el entorno de desarrollo, de acuerdo a la guía facilitada por la cátedra. Trabajamos con la distribución Ubuntu (basada en Debian) que se puede bajar libremente de http://www.ubuntu.com/. Se realizó posteriormente la instalación de GxEmul para emular un sistema MIPS; se utilizó el proporcionado por la cátedra, el cual traía una imagen del sistema operativo NetBSD con algunas utilidades (compilador C, ssh, editor vi, etc).

4.2. Paso 2: Implementación del programa

El programa se subdividió en dos funciones:
- main, encargada del parseo de los parámetros de entrada.
- procesarArchivo, encargada de realizar la funcionalidad del comando rev.

El programa debe ejecutarse por línea de comando, donde la salida sera también por consola.

4.2.1. Implementación de las funciones

- main Es la función principal, encargada de tomar los parámetros de entrada. Luego imprimir ayuda o versión si corresponde, o realizar llamadas sucesivas a la función procesarArchivo con el puntero a archivo que corresponda, ya sea el cual cuyo nombre se recibe como parámetro o simplemente *stdin*.

- procesarArchivo Es la función encargada de realizar la inversión de los caracteres de las líneas de los archivos de entrada. Lee secuencialmente las líneas reservando 1 byte de memoria adicional con cada caracter leído.
  Una vez en memoria se arma un segundo arreglo de caracteres donde se copia la línea invirtiendola. Finalmente se imprime por stdout esta línea invertida.
  Se libera la memoria y se lee la siguiente línea del archivo. El ciclo corta al llegar a una señal de EOF.

4.2.2. Ingreso de parámetros

El formato para invocar al programa es el siguiente:
./tp1 nombreArchivo1 nombreArchivo2 .. nombreArchivoN
Donde nombreArchivoX posee el texto al cual aplicar el comando rev.

5. Código para compilar el programa con gcc

El proyecto se debe compilar tanto en el sistema Host (en nuestro caso Ubuntu) como en el Guest (NetBSD). Se dispone del mismo compilador en ambos sistemas por lo tanto para ambos casos debemos situarnos en el directorio donde se encuentran todos los fuentes y utilizar el siguiente comando:

```
gcc -std=c99 -o tp0 tp0.c
```

Con esto se generará un archivo ejecutable, llamado tp0.

6. Ejemplos de ejecución

A continuación se mostrarán algunos ejemplos de archivos sobre los cuales se utiliza el programa realizado.
Considerando que los caracteres que vamos a tener en cuenta son de 1 byte realizaremos pruebas sobre archivos de texto similares al inglés generados aleatoreamente utilizando la herramienta proporcionada en el sitio
http://randomtextgenerator.com/

$ cat ejemplo1.txt
Turned it up should no valley cousin he.
Speaking numerous ask did horrible packages set. Ashamed herself has distant can studied mrs. Led therefore its middleton perpetual fulfilled provision frankness.
Small he drawn after among every three no. All having but you edward genius though remark one.

Perhaps far exposed age effects. Now distrusts you her delivered applauded affection out sincerity. As tolerably recommend shameless unfeeling he objection consisted.
She although cheerful perceive screened throwing met not eat distance.
Viewing hastily or written dearest elderly up weather it as. So direction so sweetness or extremity at daughters. Provided put unpacked now but bringing.

$ ./tp0 ejemplo1.txt
.eh nisuoc yellav on dluohs pu ti denruT
.ssenknarf noisivorp dellifluf lauteprep notelddim sti erofereht deL .srm deiduts nac tnatsid sah flesreh demahsA .tes segakcap elbirroh did ksa suoremun gnikaepS
.eno kramer hguoht suineg drawde uoy tub gnivah llA .on eerht yreve gnoma retfa nward eh llamS

.detsisnoc noitcejbo eh gnileefnu sselemahs dnemmocer ylbarelot sA .ytirecnis tuo noitceffa dedualppa dereviled reh uoy stsurtsid woN .stceffe ega desopxe raf spahreP
.ecnatsid tae ton tem gniworht deneercs eviecrep lufreehc hguohtla ehS
.gnignirb tub won dekcapnu tup dedivorP .srethguad ta ytimertxe ro ssenteews os noitcerid oS .sa ti rehtaew pu ylredle tseraed nettirw ro ylitsah gniweiV

$ cat ejemplo2.txt
Bepaalde ik mogelijk interest gestoken in de wisselen er. Ten dan toe kinderen uitgaven stampers verhoogd. Leeningen wat uit wassching siameezen. ondernomen af verscholen en en. Drong die weg mei ploeg tabak ook. Kan bewijs deelen dan gambir midden ceylon.

Mee geheelen wat gas kapitaal strooien kolonist mineraal. Ook erin wie maar zien weer moet ader. Aard maar nu wier de daad. Personeel levert nu om. Werkt van dit wijze buurt dagen bezet een heele.

Afstands in is cultures reiziger de. Pahang geruineerd in plotseling. Jungles bijgang te nu beperkt op vrouwen. En doelmatige om ad traliewerk bevorderen.

$ cat ejemplo2.txt |./tp0
.nolyec neddim ribmag nad neleed sjiweb naK .koo kabat geolp iem gew eid gnorD .ne ne nelohcsrev fa nemonredno .nezeemais gnihcssaw tiu taw negnineeL .dgoohrev srepmats nevagtiu nerednik eot nad neT .re nelessiw ed ni nekotseg tseretni kjilegom ki edlaapeB

.eleeh nee tezeb negad truub ezjiw tid nav tkreW .mo un trevel leenosreP .daad ed reiw un raam draA .reda teom reew neiz raam eiw nire koO .laarenim tsinolok neioorts laatipak sag taw neleeheg eeM

.neredroveb kreweilart da mo egitamleod nE .newuorv po tkrepeb un et gnagjib selgnuJ .gnilestolp ni dreeniureg gnahaP .ed regizier serutluc si ni sdnatsfA

7. Comparación con comando rev de UNIX

Realizamos las siguientes pruebas para comparar la salida generada por el Trabajo Práctico con el comando *rev* de UNIX.

$ ./tp0 ejemplo1.txt >ejemplo1.out
$ rev ejemplo1.txt >ejemplo1.out.rev
$ diff ejemplo1.out ejemplo1.out.rev

$ cat ejemplo2.txt |./tp0 >ejemplo2.out
$ rev ejemplo2.txt >ejemplo2.out.rev
$ diff ejemplo2.out ejemplo2.out.rev

Para todos los casos no se obtuvo salida por consola por lo que ambos archivos de salida son iguales.

8. Conclusión

En el desarrollo del trabajo práctico nos familiarizamos con la infraestructura básica que se utilizará durante la cursada para el estudio del código assembly MIPS32.
Es importante notar la similitud a nivel usuario de los sistemas operativos basados en UNIX, cuando sin embargo se encuentran corriendo sobre arquitecturas diferentes. Esto es apreciable cuando se genera el código assembly en NetBSD que corre sobre la arquitectura MIPS.
En cuanto al código realizado se procuró que cumpla con la funcionalidad básica exigida por el enunciado, leyendo archivos tanto por parámetro como por entrada standard. El programa emite código de retorno 1 para todos los casos de error.

9. Bibliografía

- Material de la cátedra
  Se puede encontrar en el grupo Yahoo:
  http://groups.yahoo.com/neo/groups/orga-comp/files

10. Código Fuente

10.1. tp0.c

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

#define VERSION 3

// Definicion de funcines

int procesarArchivo(FILE* fd);

// Funcion principal
int main(int argc, char** argv) {
  int status = 0;

  if (argc == 2) {
    if (strcmp(argv[1], "-V") == 0) {
      printf("Version: %d\n", VERSION);
      return 0;
    }
    if (strcmp(argv[1], "-h") == 0) {
      printf("Comandos y argumentos disponibles:\n");
      printf("-V Version del programa\n");
      printf("[File...] (Archivo/s de entrada)\n");
      printf(
          "En caso de no pasar archivos se toma la entrada estandar\n");
      printf("Cada linea se cuenta hasta un enter\n");
      printf(
          "Para finalizar el programa estando con la entrada estandar");
      printf(" pulsar 'ctrl+d' para un correcto cierre del mismo");
      // Y cualquier otra cosa que se quiera agregar
      return 0;
    }
  }
  if (argc == 1) {
    status = procesarArchivo(stdin);
  } else {
    int arch = 1;
    FILE* entrada;
    while (arch < argc) {
      entrada = fopen(argv[arch], "r");
      if (!entrada) {
        fprintf(stderr, "No se pudo abrir el archivo: %s\n",
            argv[arch]);
        return 1;
      }
      status = procesarArchivo(entrada);
      if (status)
        return status;
      fclose(entrada);
      entrada = NULL;
      arch++;
    }
  }
  return status;

}

// Procesa el archivo de entrada (puede ser stdin)
```

```
// Invierte las lineas del archivo y las imprime por stdout
// En caso satisfactorio devuelve 0, distinto de 0 en otros casos
int procesarArchivo(FILE* fd) {
  int nextChar = fgetc(fd);
  size_t sizeOfChar = sizeof(unsigned char);
  unsigned char* line = malloc(sizeOfChar);
  if (!line) {
    fprintf(stderr, "Problema al querer alocar memoria\n");
    return -1;
  }
  int lineWidth = 0;
  int result = 0;
  int aux_special;
  while (nextChar != EOF) {
    if (nextChar > 127) { // Para solucionar tema de caracteres especiales
      lineWidth += 2;
      line = (unsigned char*) realloc(line, lineWidth * sizeOfChar);
      if (!line) {
        fprintf(stderr, "Problema al querer alocar memoria\n");
        return -1;
      }
      aux_special = nextChar;
      nextChar = fgetc(fd);
      line[lineWidth - 2] = (unsigned char) nextChar;
      line[lineWidth - 1] = (unsigned char) aux_special;
      nextChar = fgetc(fd);
      continue;
    }
    if (nextChar != '\n') {
      lineWidth++;
      line = (unsigned char*) realloc(line, lineWidth * sizeOfChar);
      if (!line) {
        fprintf(stderr, "Problema al querer alocar memoria\n");
        return -1;
      }
      line[lineWidth - 1] = (unsigned char) nextChar;
    } else {
      unsigned char* inverseLine = (unsigned char*) malloc(
          (lineWidth + 1) * sizeOfChar);
      for (int pos = 0; pos < lineWidth; pos++) {
        inverseLine[pos] = line[(lineWidth - 1) - pos];
      }
      inverseLine[lineWidth] = '\n';
      int outStatus = fwrite(inverseLine, sizeOfChar, lineWidth + 1,
          stdout);
      if (outStatus != lineWidth + 1) {
        fprintf(stderr, "Error al escribir en salida\n");
        result = 2;
      }
      free(inverseLine);
      lineWidth = 0;
    }
    nextChar = fgetc(fd);
  }
  free(line);
  return result;
}
```

11. Código MIPS generado por el compilador

11.1. tp0.s

```
  .file 1 "tp0.c"
  .section .mdebug.abi32
  .previous
  .abicalls
  .rdata
  .align  2
$LC0:
  .ascii  "-V\000"
  .align  2
$LC1:
  .ascii  "Version: %d\n\000"
  .align  2
$LC2:
  .ascii  "-h\000"
  .align  2
$LC3:
  .ascii  "Comandos y argumentos disponibles:\n\000"
  .align  2
$LC4:
  .ascii  "-V Version del programa\n\000"
  .align  2
$LC5:
  .ascii  "[File...] (Archivo/s de entrada)\n\000"
  .align  2
$LC6:
  .ascii  "En caso de no pasar archivos se toma la entrada estandar"
  .ascii  "\n\000"
  .align  2
$LC7:
  .ascii  "Cada linea se cuenta hasta un enter\n\000"
  .align  2
$LC8:
  .ascii  "Para finalizar el programa estando con la entrada estand"
  .ascii  "ar\000"
  .align  2
$LC9:
  .ascii  " pulsar 'ctrl+d' para un correcto cierre del mismo\000"
  .align  2
$LC10:
  .ascii  "r\000"
  .align  2
$LC11:
  .ascii  "No se pudo abrir el archivo: %s\n\000"
  .text
  .align  2
  .globl  main
  .ent  main
main:
  .frame  $fp,56,$31    # vars= 16, regs= 3/0, args= 16, extra= 8
  .mask 0xd0000000,-8
  .fmask  0x00000000,0
  .set  noreorder
  .cpload $25
  .set  reorder
  subu  $sp,$sp,56
  .cprestore 16
  sw  $31,48($sp)
  sw  $fp,44($sp)
```

```
  sw   $28,40($sp)
  move   $fp,$sp
  sw   $4,56($fp)
  sw   $5,60($fp)
  sw   $0,24($fp)
  lw   $3,56($fp)
  li   $2,2        # 0x2
  bne  $3,$2,$L18
  lw   $2,60($fp)
  addu   $2,$2,4
  lw   $4,0($2)
  la   $5,$LC0
  la   $25,strcmp
  jal  $31,$25
  bne  $2,$0,$L19
  la   $4,$LC1
  li   $5,3        # 0x3
  la   $25,printf
  jal  $31,$25
  sw   $0,36($fp)
  b $L17
$L19:
  lw   $2,60($fp)
  addu   $2,$2,4
  lw   $4,0($2)
  la   $5,$LC2
  la   $25,strcmp
  jal  $31,$25
  bne  $2,$0,$L18
  la   $4,$LC3
  la   $25,printf
  jal  $31,$25
  la   $4,$LC4
  la   $25,printf
  jal  $31,$25
  la   $4,$LC5
  la   $25,printf
  jal  $31,$25
  la   $4,$LC6
  la   $25,printf
  jal  $31,$25
  la   $4,$LC7
  la   $25,printf
  jal  $31,$25
  la   $4,$LC8
  la   $25,printf
  jal  $31,$25
  la   $4,$LC9
  la   $25,printf
  jal  $31,$25
  sw   $0,36($fp)
  b $L17
$L18:
  lw   $3,56($fp)
  li   $2,1        # 0x1
  bne  $3,$2,$L21
  la   $4,__sF
  la   $25,procesarArchivo
  jal  $31,$25
  sw   $2,24($fp)
  b $L22
$L21:
```

```
  li   $2,1        # 0x1
  sw   $2,28($fp)
$L23:
  lw   $2,28($fp)
  lw   $3,56($fp)
  slt  $2,$2,$3
  bne  $2,$0,$L25
  b $L22
$L25:
  lw   $2,28($fp)
  sll  $3,$2,2
  lw   $2,60($fp)
  addu   $2,$3,$2
  lw   $4,0($2)
  la   $5,$LC10
  la   $25,fopen
  jal  $31,$25
  sw   $2,32($fp)
  lw   $2,32($fp)
  bne  $2,$0,$L26
  lw   $2,28($fp)
  sll  $3,$2,2
  lw   $2,60($fp)
  addu   $2,$3,$2
  la   $4,__sF+176
  la   $5,$LC11
  lw   $6,0($2)
  la   $25,fprintf
  jal  $31,$25
  li   $2,1        # 0x1
  sw   $2,36($fp)
  b $L17
$L26:
  lw   $4,32($fp)
  la   $25,procesarArchivo
  jal  $31,$25
  sw   $2,24($fp)
  lw   $2,24($fp)
  beq  $2,$0,$L27
  lw   $2,24($fp)
  sw   $2,36($fp)
  b $L17
$L27:
  lw   $4,32($fp)
  la   $25,fclose
  jal  $31,$25
  sw   $0,32($fp)
  lw   $2,28($fp)
  addu   $2,$2,1
  sw   $2,28($fp)
  b $L23
$L22:
  lw   $2,24($fp)
  sw   $2,36($fp)
$L17:
  lw   $2,36($fp)
  move   $sp,$fp
  lw   $31,48($sp)
  lw   $fp,44($sp)
  addu   $sp,$sp,56
  j $31
  .end   main
```

```
.size main, .-main
.ident "GCC: (GNU) 3.3.3 (NetBSD nb3 20040520)"
```