Trabajo Práctico 0: infraestructura básica

66.20 Organización de las Computadoras

Nicolás Calvo, *Padrón Nro. 78.914*
nicolas.g.calvo@gmail.com
Celeste Maldonado, *Padrón Nro. 85.630*
maldonado.celeste@gmail.com
Matias Acosta, *Padrón Nro. 88.590*
matiasja@gmail.com

2do. Cuatrimestre de 2011

Facultad de Ingeniería, Universidad de Buenos Aires

Índice

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

24/08/2011

1. Objetivos

Familiarizarse con las herramientas de software que usaremos en los siguientes trabajos, implementado un programa (y su correspondiente documentación) que resuelva el problema piloto que se presentará a continuación.

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, y se valorarán aquellos escritos usando la herramienta TEX / LATEX.

4. Recursos

Usaremos el programa GXemul [1] 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]. GXemul se puede hacer correr bajo Windows, en el entorno Cygwin [2].

5. Implementación

Programa

El programa a escribir en lenguage C, es una versión minimalista del comando `join` [4] de UNIX. El mismo, realizará la unión de dos archivos según la primer palabra/campo de cada archivo que será tomado como clave de unión. Si se pasa sólo uno de los archivos a unir, se leerá de la entrada estándar. Se deberán implementar las opciones:

- `-V, --version`
- `-h, --help`
- `-i, --ignore-case`

En caso de existir errores, estos deben ser impresos por *stderr*.

Ejemplo

Usamos la opción -h para ver el mensaje de ayuda:

```
tp0 -h

Usage: join [OPTION]... FILE1 FILE2
For each pair of input lines with identical join fields, write a line to
standard output.  The default join field is the first, delimited
by whitespace.  When FILE1 or FILE2 (not both) is -, read standard input.

  -i, --ignore-case  ignore differences in case when comparing fields
  -h, --help         display this help and exit
  -v, --version      output version information and exit

Important: FILE1 and FILE2 must be sorted on the join fields.
E.g., use ‘sort -k 1b,1’ if ‘join’ has no options.
Note, comparisons honor the rules specified by ‘LC_COLLATE’.
If the input is not sorted and some lines cannot be joined, a
warning message will be given.
```

Ejemplo de una ejecución:

```
$ cat ApellidoNombre
1 Djokovic, Novak
2 Nadal, Rafael
3 Federer, Roger
4 Murray, Andy
5 Ferrer, David
6 Soderling, Robin
7 Monfils, Gael
8 Fish, Mardy
```

```
9 Berdych, Tomas
10 Almagro, Nicolas

$ cat Puntaje
1 13,920
2 11,420
3 8,380
4 6,535
5 4,200
6 4,145
7 3,165
8 2,820
9 2,690
10 2,380

$ tp0 ApellidoNombre Puntaje
1 Djokovic, Novak 13,920
2 Nadal, Rafael 11,420
3 Federer, Roger 8,380
4 Murray, Andy 6,535
5 Ferrer, David 4,200
6 Soderling, Robin 4,145
7 Monfils, Gael 3,165
8 Fish, Mardy 2,820
9 Berdych, Tomas 2,690
10 Almagro, Nicolas 2,380

$
```

El programa debe tener como salida, la misma información que el comando `join` de UNIX. Se debe respetar la presentación de esta tal cual la realiza el comando. Además el programa debe soportar el orden, cantidad y disposición de los parámetros tal cual lo soporta el comando original de UNIX.

Para más información consultar el manual del comando `join` y un ejemplo de uso en [5]

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[1]) y la versión para Linux (en cualquiera de sus distribuciones) para correr el simulador.

2. Introducción

Para la realización del trabajo práctico fue necesario simular un sistema operativo (NetBSD) que utiliza un procesador MIPS. En el simulador utilizamos el programa GXemul, que permite simular el entorno necesario para producir el código, compilarlo, ejecutarlo y obtener el código MIPS32 generado por el compilador.

3. Programa

El programa, a escribir en lenguaje C, es una versión minimalista del comando join de UNIX. El mismo, realizará la unión de dos archivos según la primer palabra/campo de cada archivo que será tomado como clave de unión. Si se pasa sólo uno de los archivos a unir, se leerá de la entrada estandar.

3.1. Diseño e implementación

Proceso y salida Primero se verifican los parámetros utilizados para llamar el programa. En caso de encontrarse un error en alguno de los argumentos de entrada se reporta mediante un mensaje de error. El formato del archivo de entrada es de texto.

Manejo de errores El manejo de errores se realiza mediante el stream stderr, en caso de error se muestra una leyenda que se corresponde con cada caso.

3.2. Generación

Para generar el binario hay que ejecutar el siguiente comando:

```
gcc -Wall -lm -O0 -o tp0 tp0.c
```

Para compilar el código a MIPS se utiliza:

```
gcc -Wall -O0 -S -mrnames tp0.c
```

- Wall: activa todos los mensajes de warning.
- O: indica el nivel de optimizacion, en este caso no queremos que el compilador optimice el programa por lo que ponemos nivel 0.
- o: genera el archivo de salida.

3.3. Uso y comandos

La entrada del programa serán los dos archivos a unir. En caso de no especificarse uno, el programa leerá de la entrada estandard stdin.

El programa debe tener como salida, la misma información que el comando join de UNIX. Se debe respetar la presentación de esta tal cual la realiza el comando. Además el programa debe soportar el orden, cantidad y disposición de los parámetros tal cual lo soporta el comando original de UNIX. Los mensajes de error deben indicarse via stderr.

A continuación se describen las opciones disponibles:

- "-V" o "–version": esta opción muestra la versión del programa. No recibe ningún argumento.
- "-h" o "–help": esta opción muestra un mensaje de ayuda, el cual posee las opciones que recibe el programa.
- "-i" o "–ignore-case": ignora la diferencia en la comparación de las claves de los archivos.

3.3.1. Ejemplo

A continuación se exponen varios ejemplos del uso de la aplicación.

Usamos la opción -h para ver el mensaje de ayuda:

```
tp0 -h
Usage: join [OPTION]... FILE1 FILE2
For each pair of input lines with identical join fields, write a line to standard ou
by whitespace. When FILE1 or FILE2 (not both) is -, read standard input. \\
-i, --ignore-case ignore differences in case when comparing fields
-h, --help display this help and exit
-v, --version output version information and exit
Important: FILE1 and FILE2 must be sorted on the join fields. \\
E.g., use ‘sort -k 1b,1’ if ‘join’ has no options. \\
Note, comparisons honor the rules specified by ‘‘LC_COLLATE". \\
If the input is not sorted and some lines cannot be joined, a warning message will b
```

Ejemplo de una ejecución:

```
$ cat ApellidoNombre
1 Djokovic, Novak
2 Nadal, Rafael
```

```
3 Federer, Roger
4 Murray, Andy
5 Ferrer, David
6 Soderling, Robin
7 Monfils, Gael
8 Fish, Mardy
9 Berdych, Tomas
10 Almagro, Nicolas

$ cat Puntaje
1 13,920
2 11,420
3 8,380
4 6,535
5 4,200
6 4,145
7 3,165
8 2,820
9 2,690
10 2,380

$ tp0 ApellidoNombre Puntaje
1 Djokovic, Novak 13,920
2 Nadal, Rafael 11,420
3 Federer, Roger 8,380
4 Murray, Andy 6,535
5 Ferrer, David 4,200
6 Soderling, Robin 4,145
7 Monfils, Gael 3,165
8 Fish, Mardy 2,820
9 Berdych, Tomas 2,690
10 Almagro, Nicolas 2,380

$
```

4. Pruebas (Test)

Se realizaron diferentes pruebas para poder abarcar todos los casos posibles que nos permitan determinar el buen funcionamiento del programa.

Prueba nº1

- Comando

- Salida tp0

4.1. Análisis de los resultados

Se prosigue a detallar algunas corridas del programa en el sistema NetBSD para mostrar su funcionalidad.
A través del comando:

```
.\tp0 -h
```

Se imprime por pantalla el mensaje de ayudar con detalles del uso, opciones y ejemplos de uso de la aplicación:

5. Código Fuente

5.1. tp0.c

5.2. tp0.s

```
    .file 1 "tp0.c"
    .section .mdebug.abi32
    .previous
    .abicalls
    .rdata
    .align 2
$LC0:
    .ascii "version\000"
    .align 2
$LC1:
    .ascii "help\000"
    .align 2
$LC2:
    .ascii "ignore-case\000"
    .data
    .align 2
    .type long_options.0, @object
    .size long_options.0, 64
long_options.0:
    .word $LC0
    .word 0
    .word 0
    .word 86
    .word $LC1
    .word 0
    .word 0
    .word 104
    .word $LC2
    .word 1
    .word 0
    .word 105
    .word 0
    .word 0
    .word 0
    .word 0
    .rdata
    .align 2
$LC3:
    .ascii "hVi:\000"
    .align 2
$LC4:
    .ascii "Version 1.00\n\000"
    .text
```

```
        .align 2
        .globl main
        .ent main
main:
        .frame $fp,192,$ra  # vars= 144, regs= 3/0, args= 24, extra= 8
        .mask 0xd0000000,-8
        .fmask 0x00000000,0
        .set noreorder
        .cpload $t9
        .set reorder
        subu $sp,$sp,192
        .cprestore 24
        sw $ra,184($sp)
        sw $fp,180($sp)
        sw $gp,176($sp)
        move $fp,$sp
        sw $a0,192($fp)
        sw $a1,196($fp)
        sw $zero,32($fp)
        li $v0,1    # 0x1
        sw $v0,44($fp)
        li $v0,1    # 0x1
        sw $v0,40($fp)
        li $v0,1    # 0x1
        sw $v0,36($fp)
$L18:
        addu $v0,$fp,32
        sw $v0,16($sp)
        lw $a0,192($fp)
        lw $a1,196($fp)
        la $a2,$LC3
        la $a3,long_options.0
        la $t9,getopt_long
        jal $ra,$t9
        sw $v0,48($fp)
        lw $v1,48($fp)
        li $v0,-1    # 0xffffffffffffffff
        bne $v1,$v0,$L20
        b $L19
$L20:
        lw $v0,48($fp)
        sw $v0,172($fp)
        li $v0,104    # 0x68
        lw $v1,172($fp)
        beq $v1,$v0,$L22
        lw $v1,172($fp)
        slt $v0,$v1,105
        beq $v0,$zero,$L28
        li $v0,63    # 0x3f
```

```
    lw $v1,172($fp)
    beq $v1,$v0,$L25
    li $v0,86    # 0x56
    lw $v1,172($fp)
    beq $v1,$v0,$L23
    b $L18
$L28:
    li $v0,105    # 0x69
    lw $v1,172($fp)
    beq $v1,$v0,$L24
    b $L18
$L22:
    sw $zero,36($fp)
    b $L18
$L23:
    sw $zero,40($fp)
    b $L18
$L24:
    sw $zero,44($fp)
    b $L18
$L25:
    li $v0,1    # 0x1
    sw $v0,168($fp)
    b $L17
$L19:
    lw $v0,40($fp)
    bne $v0,$zero,$L29
    la $a0,$LC4
    la $t9,printf
    jal $ra,$t9
    sw $zero,168($fp)
    b $L17
$L29:
    lw $v0,36($fp)
    bne $v0,$zero,$L30
    la $t9,print_help
    jal $ra,$t9
    sw $zero,168($fp)
    b $L17
$L30:
    lw $v0,44($fp)
    bne $v0,$zero,$L31
    sw $zero,168($fp)
    b $L17
$L31:
    lw $v0,192($fp)
    slt $v0,$v0,2
    bne $v0,$zero,$L32
    lw $v0,196($fp)
```

```
    addu $v0,$v0,4
    lw $v0,0($v0)
    sw $v0,52($fp)
    lw $v0,192($fp)
    slt $v0,$v0,3
    bne $v0,$zero,$L33
    lw $v0,196($fp)
    addu $v0,$v0,8
    lw $v0,0($v0)
    sw $v0,56($fp)
    b $L34
$L33:
    addu $v0,$fp,64
    move $a0,$v0
    li $a1,100    # 0x64
    la $a2,__sF
    la $t9,fgets
    jal $ra,$t9
    addu $v0,$fp,64
    move $a0,$v0
    la $t9,limpiar
    jal $ra,$t9
    addu $v0,$fp,64
    sw $v0,56($fp)
$L34:
    lw $a0,52($fp)
    lw $a1,56($fp)
    la $t9,funcionJoin
    jal $ra,$t9
$L32:
    sw $zero,168($fp)
$L17:
    lw $v0,168($fp)
    move $sp,$fp
    lw $ra,184($sp)
    lw $fp,180($sp)
    addu $sp,$sp,192
    j $ra
    .end main
    .size main, .-main
    .rdata
    .align 2
$LC5:
    .ascii "\n"
    .ascii "Usage: join [OPTION]... FILE1 FILE2\n\000"
    .align 2
$LC6:
    .ascii "For each pair of input lines with identical join fields,"
    .ascii " write a line to\n\000"
```

```
    .align 2
$LC7:
    .ascii "standard output. The default join field is the first, de"
    .ascii "limited\n\000"
    .align 2
$LC8:
    .ascii "by whitespace. When FILE1 or FILE2 (not both) is -, read"
    .ascii " standard input.\n\000"
    .align 2
$LC9:
    .ascii "  -i, --ignore-case ignore differences in case when com"
    .ascii "paring fields\n\000"
    .align 2
$LC10:
    .ascii "  -h, --help display this help and exit\n\000"
    .align 2
$LC11:
    .ascii "  -v, --version output version information and exit"
    .ascii "\n\000"
    .align 2
$LC12:
    .ascii "Important: FILE1 and FILE2 must be sorted on the join fi"
    .ascii "elds.\n\000"
    .align 2
$LC13:
    .ascii "E.g., use \342\200\230sort -k 1b,1\342\200\231 if \342\200"
    .ascii "\230join\342\200\231 has no options.\n\000"
    .align 2
$LC14:
    .ascii "Note, comparisons honor the rules specified by \342\200\230"
    .ascii "LC_COLLATE\342\200\231.\n\000"
    .align 2
$LC15:
    .ascii "If the input is not sorted and some lines cannot be join"
    .ascii "ed, a\n\000"
    .align 2
$LC16:
    .ascii "warning message will be given.\n\n\000"
    .text
    .align 2
    .globl print_help
    .ent print_help
print_help:
    .frame $fp,40,$ra  # vars= 0, regs= 3/0, args= 16, extra= 8
    .mask 0xd0000000,-8
    .fmask 0x00000000,0
    .set noreorder
    .cpload $t9
    .set reorder
```

```
subu $sp,$sp,40
.cprestore 16
sw $ra,32($sp)
sw $fp,28($sp)
sw $gp,24($sp)
move $fp,$sp
la $a0,$LC5
la $t9,printf
jal $ra,$t9
la $a0,$LC6
la $t9,printf
jal $ra,$t9
la $a0,$LC7
la $t9,printf
jal $ra,$t9
la $a0,$LC8
la $t9,printf
jal $ra,$t9
la $a0,$LC9
la $t9,printf
jal $ra,$t9
la $a0,$LC10
la $t9,printf
jal $ra,$t9
la $a0,$LC11
la $t9,printf
jal $ra,$t9
la $a0,$LC12
la $t9,printf
jal $ra,$t9
la $a0,$LC13
la $t9,printf
jal $ra,$t9
la $a0,$LC14
la $t9,printf
jal $ra,$t9
la $a0,$LC15
la $t9,printf
jal $ra,$t9
la $a0,$LC16
la $t9,printf
jal $ra,$t9
move $sp,$fp
lw $ra,32($sp)
lw $fp,28($sp)
addu $sp,$sp,40
j $ra
.end print_help
.size print_help, .-print_help
```

```
    .align 2
    .globl limpiar
    .ent limpiar
limpiar:
    .frame $fp,48,$ra  # vars= 8, regs= 3/0, args= 16, extra= 8
    .mask 0xd0000000,-8
    .fmask 0x00000000,0
    .set noreorder
    .cpload $t9
    .set reorder
    subu $sp,$sp,48
    .cprestore 16
    sw $ra,40($sp)
    sw $fp,36($sp)
    sw $gp,32($sp)
    move $fp,$sp
    sw $a0,48($fp)
    lw $a0,48($fp)
    li $a1,10   # 0xa
    la $t9,strchr
    jal $ra,$t9
    sw $v0,24($fp)
    lw $v0,24($fp)
    beq $v0,$zero,$L36
    lw $v0,24($fp)
    sb $zero,0($v0)
$L36:
    move $sp,$fp
    lw $ra,40($sp)
    lw $fp,36($sp)
    addu $sp,$sp,48
    j $ra
    .end limpiar
    .size limpiar, .-limpiar
    .rdata
    .align 2
$LC17:
    .ascii "r\000"
    .align 2
$LC18:
    .ascii "Error al abrir  %s \n\000"
    .align 2
$LC19:
    .ascii " %c  %s  %s \n\000"
    .align 2
$LC20:
    .ascii "No se encontro la clave en el segundo archivo \n\000"
    .text
    .align 2
```

```
	.globl funcionJoin
	.ent funcionJoin
funcionJoin:
	.frame $fp,384,$ra  # vars= 344, regs= 3/0, args= 16, extra= 8
	.mask 0xd0000000,-8
	.fmask 0x00000000,0
	.set noreorder
	.cpload $t9
	.set reorder
	subu $sp,$sp,384
	.cprestore 16
	sw $ra,376($sp)
	sw $fp,372($sp)
	sw $gp,368($sp)
	move $fp,$sp
	sw $a0,384($fp)
	sw $a1,388($fp)
	lw $a0,384($fp)
	la $a1,$LC17
	la $t9,fopen
	jal $ra,$t9
	sw $v0,24($fp)
	lw $a0,388($fp)
	la $a1,$LC17
	la $t9,fopen
	jal $ra,$t9
	sw $v0,28($fp)
	li $v0,1	# 0x1
	sw $v0,352($fp)
	li $v0,1	# 0x1
	sw $v0,356($fp)
	sw $zero,360($fp)
	lw $v0,24($fp)
	bne $v0,$zero,$L39
	la $a0,__sF+176
	la $a1,$LC18
	lw $a2,384($fp)
	la $t9,fprintf
	jal $ra,$t9
	li $v0,1	# 0x1
	sw $v0,360($fp)
$L39:
	lw $v0,28($fp)
	bne $v0,$zero,$L40
	la $a0,__sF+176
	la $a1,$LC18
	lw $a2,388($fp)
	la $t9,fprintf
	jal $ra,$t9
```

```
    li $v0,1    # 0x1
    sw $v0,360($fp)
$L40:
    lw $v0,360($fp)
    bne $v0,$zero,$L38
    lw $a0,24($fp)
    la $t9,fgetc
    jal $ra,$t9
    sb $v0,32($fp)
$L42:
    lw $v0,24($fp)
    lhu $v0,12($v0)
    srl $v0,$v0,5
    xori $v0,$v0,0x1
    andi $v0,$v0,0x1
    beq $v0,$zero,$L43
    lw $v1,356($fp)
    li $v0,1    # 0x1
    beq $v1,$v0,$L44
    b $L43
$L44:
    addu $v0,$fp,144
    move $a0,$v0
    li $a1,100    # 0x64
    lw $a2,24($fp)
    la $t9,fgets
    jal $ra,$t9
    addu $v0,$fp,144
    move $a0,$v0
    la $t9,limpiar
    jal $ra,$t9
$L46:
    lw $v1,352($fp)
    li $v0,1    # 0x1
    bne $v1,$v0,$L47
    lw $v0,28($fp)
    lhu $v0,12($v0)
    srl $v0,$v0,5
    xori $v0,$v0,0x1
    andi $v0,$v0,0x1
    bne $v0,$zero,$L48
    b $L47
$L48:
    lw $a0,28($fp)
    la $t9,fgetc
    jal $ra,$t9
    sb $v0,33($fp)
    lb $v1,32($fp)
    lb $v0,33($fp)
```

```
    bne $v1,$v0,$L50
    sw $zero,352($fp)
    addu $v0,$fp,248
    move $a0,$v0
    li $a1,100   # 0x64
    lw $a2,28($fp)
    la $t9,fgets
    jal $ra,$t9
    addu $v0,$fp,248
    move $a0,$v0
    la $t9,limpiar
    jal $ra,$t9
    b $L46
$L50:
    addu $v0,$fp,40
    move $a0,$v0
    li $a1,100   # 0x64
    lw $a2,28($fp)
    la $t9,fgets
    jal $ra,$t9
    b $L46
$L47:
    lw $v0,352($fp)
    bne $v0,$zero,$L52
    lb $v0,32($fp)
    addu $v1,$fp,144
    addu $a3,$fp,248
    la $a0,$LC19
    move $a1,$v0
    move $a2,$v1
    la $t9,printf
    jal $ra,$t9
    li $v0,1   # 0x1
    sw $v0,352($fp)
    lw $a0,24($fp)
    la $t9,fgetc
    jal $ra,$t9
    sb $v0,32($fp)
    lw $a0,28($fp)
    la $t9,rewind
    jal $ra,$t9
    b $L42
$L52:
    sw $zero,356($fp)
    b $L42
$L43:
    lw $v0,356($fp)
    bne $v0,$zero,$L54
    la $a0,__sF+176
```

```
    la $a1,$LC20
    la $t9,fprintf
    jal $ra,$t9
$L54:
    lw $a0,24($fp)
    la $t9,fclose
    jal $ra,$t9
    lw $a0,28($fp)
    la $t9,fclose
    jal $ra,$t9
$L38:
    move $sp,$fp
    lw $ra,376($sp)
    lw $fp,372($sp)
    addu $sp,$sp,384
    j $ra
    .end funcionJoin
    .size funcionJoin, .−funcionJoin
    .ident "GCC: (GNU) 3.3.3 (NetBSD nb3 20040520)"
```

6. Conclusión

A través del trabajo práctico logramos familizarnos con herramientas del entorno *Gnu/Linux*. Entre estas herramientas se pueden descatar: *ssh* (Secure Shell), *scp* (Secure Copy), algunos comandos básicos de linux, la utilización de *pipeline* para el flujo de datos y el entorno *netBSD*.