Universidad De Buenos Aires

Facultad De Ingeniería

66.20 Organización De Computadoras

Trabajo Práctico 0

Integrantes:
Gonzalo Beviglia - 93144
Federico Quevedo - 93159
Damián Manoff - 93169

10 de Septiembre de 2013

Índice

1. Diseño e implementación

Uno de los principales limitantes de las soluciones que pudimos implementar era el largo del archivo a invertir, ya que este podia llegar a ocupar mucho lugar en la memoria. Buscamos mejorar esta situación leyendo e invirtiendo de a un archivo por vez, para evitar tener mas de uno en memoria. También evitamos tener que declarar otro espacio de memoria del mismo tamaño del archivo original, donde iria la cadena invertida, haciendo una inversion in situ de la cadena, es decir, sobre la misma memoria reservada a donde se cargo el archivo. Esto ademas de ahorrarnos duplicar la memoria usada nos ahorra el tiempo de reservar la misma. Sobre la cadena original se realizan swaps espejados hasta llegar a la cadena invertida.

2. Performance

La performance se evaluó invirtiendo el libro "El Príncipe" de Nicolás Maquiavelo, y se comparó la performance del comando realizado para este trabajo práctico con la del comando unix *rev*. El tamaño de dicho texto en formato de texto plano es de 305864 bytes (298KB).

Los tiempos se midieron utilizando el comando Unix *time*.

2.1. Tiempo *ownRev*

```
real 0m0.539s

user 0m0.008s

sys 0m0.016s
```

2.2. Tiempo Unix *rev*

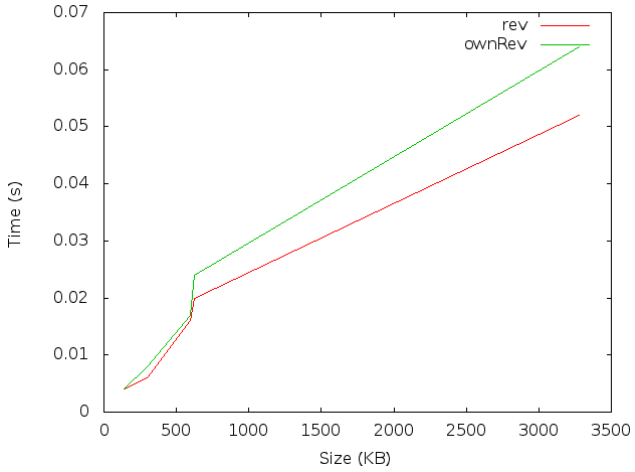
```
real 0m0.540s

user 0m0.012s

sys 0m0.024s
```

2.3. Comparación grafica

Los textos utilizados para esta comparación fueron:

- The Divine Comedy - Dante Alighieri (627KB)
- Adventures of Huckleberry Finn - Mark Twain (596KB)
- Les Miserables - Victor Hugo (3.2MB)
- Metamorphosis - Franz Kafka (139KB)
- The Prince - Nicolo Machiavelli (299KB)

Figura 1: Comparacion entre rev de Unix y nuestro “ownRev”



3. Compilación del programa

Para el compilado del programa hicimos el siguiente makefile:

```
CFLAGS=-Wall -pedantic -std=c99 -g
MEMFLAGS=valgrind --leak-check=full --track-origins=yes -v
CC=gcc
INPUT=ownRev.c
MIDDLEFILE=ownRev.o
EXEC=ownRev
TESTSCRIPT=TestFiles/tests.sh

all: $(EXEC) clean

.SILENT:
$(MIDDLEFILE):
$(CC) $(CFLAGS) $(INPUT) -c

.SILENT:
$(EXEC): $(MIDDLEFILE)
$(CC) $(CFLAGS) $(MIDDLEFILE) -o $(EXEC)

.SILENT:
run:
./$(EXEC)

.SILENT:
runTests:
./$(TESTSCRIPT)

.SILENT:
memCheck:
$(MEMFLAGS) ./$(EXEC) TestFiles/test TestFiles/test1 TestFiles/test2

.SILENT:
clean:
rm *.o

.SILENT:
```

```
cleanExec:
rm $(EXEC)
```

La ejecución normal de este make file produce el archivo ejecutable y ademas elimina los intermediarios.

Se puede tambien llamar pasando como parametro el nombre del archivo intermediario para generarlo, o el nombre del ejecutable, que realizara lo mismo que la ejecución por defecto pero sin eliminar el intermediario.

Para corroborar que no se estuviera perdiendo memoria tambien incluimos el parametro *memCheck* que corre el programa con valgrind informando si hubo o no alguna perdida.

Desde el mismo makefile tambien incluimos la posibilidad de correr las pruebas, y por ultimo, la de eliminar los archivos generados, tanto intermediarios como programa final.

4. Pruebas

Para las pruebas hicimos el siguiente script de bash:

```
#!/bin/bash
FILES="TestFiles/test
TestFiles/test1
TestFiles/test2"

for f in $FILES
do
   ./ownRev $f > auxRev
   ./ownRev auxRev > doubleRev
   DIFF=$(diff $f doubleRev)
   if [ "$DIFF" != "" ]
   then
      echo "[ERROR] $f was not reversed correctly"
   else
      echo "[OK] $f was reversed correctly"
   fi
   rm auxRev
   rm doubleRev
done
```

Se toman lineas de un archivo, en este caso llamado test, y se las invierte dos veces. Si la inversion se realizo correctamente no deberia haber diferencias entre el texto introducido y el resultante.

5. Código Assembly MIPS

A continuación se detallará el código assembly para la arquitectura MIPS de nuestro programa en C

5.1. Código assembly

```
.file 1 "tp0v2.c"
.section .mdebug.abi32
.previous
.abicalls
.rdata
.align 2
$LC0:
.ascii "-v\000"
.align 2
$LC1:
.ascii "--version\000"
```

```
.align 2
$LC2:
.ascii "Version 1.0.0\n\000"
.align 2
$LC3:
.ascii "-h\000"
.align 2
$LC4:
.ascii "--help\000"
.align 2
$LC5:
.ascii "Usage\n\000"
.text
.align 2
.globl checkOption
.ent checkOption
checkOption:
.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)
la $a1,$LC0
la $t9,strcmp
jal $ra,$t9
beq $v0,$zero,$L19
lw $a0,48($fp)
la $a1,$LC1
la $t9,strcmp
jal $ra,$t9
bne $v0,$zero,$L18
$L19:
la $a0,$LC2
la $t9,printf
jal $ra,$t9
li $v0,1 # 0x1
sw $v0,24($fp)
b $L17
$L18:
lw $a0,48($fp)
la $a1,$LC3
la $t9,strcmp
jal $ra,$t9
beq $v0,$zero,$L22
lw $a0,48($fp)
la $a1,$LC4
la $t9,strcmp
```

```
jal $ra,$t9
bne $v0,$zero,$L20
$L22:
la $a0,$LC5
la $t9,printf
jal $ra,$t9
li $v0,1 # 0x1
sw $v0,24($fp)
b $L17
$L20:
sw $zero,24($fp)
$L17:
lw $v0,24($fp)
move $sp,$fp
lw $ra,40($sp)
lw $fp,36($sp)
addu $sp,$sp,48
j $ra
.end checkOption
.size checkOption, .-checkOption
.align 2
.globl swapChars
.ent swapChars
swapChars:
.frame $fp,24,$ra # vars= 8, regs= 2/0, args= 0, extra= 8
.mask 0x50000000,-4
.fmask 0x00000000,0
.set noreorder
.cpload $t9
.set reorder
subu $sp,$sp,24
.cprestore 0
sw $fp,20($sp)
sw $gp,16($sp)
move $fp,$sp
sw $a0,24($fp)
sw $a1,28($fp)
sw $a2,32($fp)
lw $v1,28($fp)
lw $v0,32($fp)
bne $v1,$v0,$L24
b $L23
$L24:
lw $v1,24($fp)
lw $v0,32($fp)
addu $v0,$v1,$v0
lbu $v0,0($v0)
sb $v0,8($fp)
lw $v1,24($fp)
lw $v0,32($fp)
addu $a0,$v1,$v0
lw $v1,24($fp)
lw $v0,28($fp)
addu $v0,$v1,$v0
lbu $v0,0($v0)
sb $v0,0($a0)
```

```
lw $v1,24($fp)
lw $v0,28($fp)
addu $v1,$v1,$v0
lbu $v0,8($fp)
sb $v0,0($v1)
$L23:
move $sp,$fp
lw $fp,20($sp)
addu $sp,$sp,24
j $ra
.end swapChars
.size swapChars, .-swapChars
.align 2
.globl reverseString
.ent reverseString
reverseString:
.frame $fp,56,$ra # vars= 16, regs= 3/0, args= 16, extra= 8
.mask 0xd0000000,-8
.fmask 0x00000000,0
.set noreorder
.cpload $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 $a0,56($fp)
la $t9,strlen
jal $ra,$t9
sw $v0,24($fp)
lw $v0,24($fp)
addu $v0,$v0,1
move $a0,$v0
la $t9,malloc
jal $ra,$t9
sw $v0,28($fp)
lw $a0,28($fp)
lw $a1,56($fp)
la $t9,strcpy
jal $ra,$t9
sw $zero,32($fp)
lw $v0,24($fp)
addu $v0,$v0,-2
sw $v0,36($fp)
$L26:
lw $v0,32($fp)
lw $v1,36($fp)
slt $v0,$v0,$v1
bne $v0,$zero,$L28
b $L27
$L28:
addu $v1,$fp,32
lw $v0,0($v1)
```

```
move $a1,$v0
addu $v0,$v0,1
sw $v0,0($v1)
addu $v1,$fp,36
lw $v0,0($v1)
move $a2,$v0
addu $v0,$v0,-1
sw $v0,0($v1)
lw $a0,28($fp)
la $t9,swapChars
jal $ra,$t9
b $L26
$L27:
lw $v0,28($fp)
move $sp,$fp
lw $ra,48($sp)
lw $fp,44($sp)
addu $sp,$sp,56
j $ra
.end reverseString
.size reverseString, .-reverseString
.align 2
.globl readFromFile
.ent readFromFile
readFromFile:
.frame $fp,64,$ra # vars= 24, regs= 3/0, args= 16, extra= 8
.mask 0xd0000000,-8
.fmask 0x00000000,0
.set noreorder
.cpload $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)
li $v0,30 # 0x1e
sw $v0,24($fp)
lw $v0,24($fp)
addu $v0,$v0,2
move $a0,$v0
la $t9,malloc
jal $ra,$t9
sw $v0,28($fp)
sw $zero,32($fp)
sb $zero,36($fp)
sw $zero,40($fp)
$L30:
lw $a0,64($fp)
la $t9,fgetc
jal $ra,$t9
sb $v0,36($fp)
lbu $v0,36($fp)
sll $v0,$v0,24
```

```
sra $v1,$v0,24
li $v0,-1 # 0xffffffffffffffff
bne $v1,$v0,$L32
b $L31
$L32:
lw $v0,40($fp)
addu $v0,$v0,1
sw $v0,40($fp)
lw $v0,40($fp)
addu $v1,$v0,1
lw $v0,24($fp)
bne $v1,$v0,$L33
lw $v0,24($fp)
sll $v0,$v0,1
sw $v0,24($fp)
lw $a0,28($fp)
lw $a1,24($fp)
la $t9,realloc
jal $ra,$t9
sw $v0,32($fp)
lw $v0,32($fp)
sw $v0,28($fp)
$L33:
lw $v1,28($fp)
lw $v0,40($fp)
addu $v0,$v1,$v0
addu $v1,$v0,-1
lbu $v0,36($fp)
sb $v0,0($v1)
lb $v1,36($fp)
li $v0,10 # 0xa
bne $v1,$v0,$L30
$L31:
addu $a1,$fp,40
lw $v1,0($a1)
move $a0,$v1
lw $v0,28($fp)
addu $v0,$a0,$v0
sb $zero,0($v0)
addu $v1,$v1,1
sw $v1,0($a1)
lw $v1,40($fp)
li $v0,1 # 0x1
bne $v1,$v0,$L35
lw $a0,28($fp)
la $t9,free
jal $ra,$t9
sw $zero,44($fp)
b $L29
$L35:
lw $a0,28($fp)
lw $a1,40($fp)
la $t9,realloc
jal $ra,$t9
sw $v0,32($fp)
lw $v0,32($fp)
```

```
sw $v0,44($fp)
$L29:
lw $v0,44($fp)
move $sp,$fp
lw $ra,56($sp)
lw $fp,52($sp)
addu $sp,$sp,64
j $ra
.end readFromFile
.size readFromFile, .-readFromFile
.rdata
.align 2
$LC6:
.ascii "%s\000"
.text
.align 2
.globl reverseFile
.ent reverseFile
reverseFile:
.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)
sw $zero,24($fp)
lw $a0,48($fp)
la $t9,readFromFile
jal $ra,$t9
sw $v0,28($fp)
$L37:
lw $v0,28($fp)
bne $v0,$zero,$L39
b $L36
$L39:
lw $a0,28($fp)
la $t9,reverseString
jal $ra,$t9
sw $v0,24($fp)
la $a0,$LC6
lw $a1,24($fp)
la $t9,printf
jal $ra,$t9
lw $a0,28($fp)
la $t9,free
jal $ra,$t9
lw $a0,24($fp)
la $t9,free
jal $ra,$t9
```

```
lw $a0,48($fp)
la $t9,readFromFile
jal $ra,$t9
sw $v0,28($fp)
b $L37
$L36:
move $sp,$fp
lw $ra,40($sp)
lw $fp,36($sp)
addu $sp,$sp,48
j $ra
.end reverseFile
.size reverseFile, .-reverseFile
.rdata
.align 2
$LC7:
.ascii "r\000"
.text
.align 2
.globl main
.ent main
main:
.frame $fp,56,$ra # vars= 16, regs= 3/0, args= 16, extra= 8
.mask 0xd0000000,-8
.fmask 0x00000000,0
.set noreorder
.cpload $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)
sw $a1,60($fp)
sw $zero,24($fp)
lw $v1,56($fp)
li $v0,1 # 0x1
bne $v1,$v0,$L41
la $a0,__sF
la $t9,reverseFile
jal $ra,$t9
sw $zero,32($fp)
b $L40
$L41:
lw $v1,56($fp)
li $v0,2 # 0x2
bne $v1,$v0,$L42
lw $v0,60($fp)
addu $v0,$v0,4
lw $a0,0($v0)
la $t9,checkOption
jal $ra,$t9
beq $v0,$zero,$L42
sw $zero,32($fp)
```

```
b $L40
$L42:
li $v0,1 # 0x1
sw $v0,28($fp)
$L44:
lw $v0,28($fp)
lw $v1,56($fp)
sltu $v0,$v0,$v1
bne $v0,$zero,$L47
b $L45
$L47:
lw $v0,28($fp)
sll $v1,$v0,2
lw $v0,60($fp)
addu $v0,$v1,$v0
lw $a0,0($v0)
la $a1,$LC7
la $t9,fopen
jal $ra,$t9
sw $v0,24($fp)
lw $a0,24($fp)
la $t9,reverseFile
jal $ra,$t9
lw $a0,24($fp)
la $t9,fclose
jal $ra,$t9
lw $v0,28($fp)
addu $v0,$v0,1
sw $v0,28($fp)
b $L44
$L45:
sw $zero,32($fp)
$L40:
lw $v0,32($fp)
move $sp,$fp
lw $ra,48($sp)
lw $fp,44($sp)
addu $sp,$sp,56
j $ra
.end main
.size main, .-main
.ident "GCC: (GNU) 3.3.3 (NetBSD nb3 20040520)"
```