

66:20 Organización de Computadoras

Trabajo Práctico 0: Infraestructura básica

Matías Acosta	88590
Nicolás Calvo	78914
Maldonado María Celeste	85630

Índice

1. Enunciado	2
2. Introducción	4
3. Programa	4
3.1. Diseño e implementación	4
4. Uso y comandos	4
4.1. Utilización	4
5. Proceso y salida	5
6. Manejo de errores	5
7. Código Fuente	6
7.1. tp0.c	6
8. Conclusión	9

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 $\text{T}_{\text{E}}\text{X}$ / $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$.

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 language 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

4. 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 estandar 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.

4.1. Utilización

Para ejecutar el programa y generar el join, deben situarse en la carpeta donde se encuentran los ejecutables, y correr el siguiente comando:

```
$ tp0 --- ---
```

5. 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.

6. 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.

7. Código Fuente

7.1. tp0.c

8. Conclusión

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