Trabajo Práctico Nº 0: Infraestructura Básica

Martinez Ariel, *Padrón Nro. 88573*
arielcorreofiuba@gmail.com.ar

Nestor Huallpa, *Padrón Nro.*
huallpa.nestor@gmail.com

Facundo Caldora, *Padrón Nro. [completar con el padron]*
[completar con mail]

1° Entrega: 08/09/2015

2do. Cuatrimestre de 2015
66.20 Organización de Computadoras
Facultad de Ingeniería, Universidad de Buenos Aires

Resumen

En el presente trabajo práctico se describirán todos los pasos y conclusiones relacionadas al desarrollo e implementación de multiplicacion de matrices de numero reales, representados en punto flotante de doble precisiòn.

Índice

1. Introducción

Completar con la introduccion...

2. Implementación

2.1. Lenguaje

Como lenguaje de implementación se eligió ANSI C [**?**] ya que el mismo permite una alta portabilidad entre diferentes plataformas. El desarrollo del programa se realizó usando un editor de texto (gedit,vim, kwrite) y compilando los archivos fuente con GCC que viene en linux (probado en Ubuntu y Fedora 12). También una parte se desarrollo usando el IDE Eclipse con su respectivo plugin para C/C++ y se comprobó que dicho programa compile con el GCC que viene dentro de NETBSD, y como adicional, también se comprobó que el mismo compile y funcione en Windows por medio del MinGW el cual es una implemetación de los compiladores GCC para la plataforma Win32 que permite migrar la capacidad de este compilador en entornos windows. La compilación en Linux y en `BSD, MIPS` se realizó con la siguiente línea de comando:
gcc -Wall -ansi -O0 -o tp0 main.c

2.2. Descripción del programa

Cuando se pasa un nombre como argumento, se verifica que dicho nombre que está haciendo referencia a un archivo, verdaderamente haga referencia a un archivo (.txt, .dat etc (Regular File)) y no a un archivo de directorio, un archivo de dispositivo etc. Una vez hecha la verificación el programa se dispone a leer cada una de las líneas desde el final hacia el principio y las vuelca por salida estándar (stdout). En caso que las lineas provengan del stdin, se utiliza un buffer de memoria dinámica que almacena todas las lineas.
La función `main` se encuentra en `tp0.c` y se encarga de interpretar las opciones y argumentos. En caso de ser una opción, como ayuda o versión, se imprime el mensaje correspondiente y finaliza la ejecución. Cuando no es una opción de ayuda o versión, se procede a procesar los datos de entrada, ya sean provenientes del stdin o de archivos. La salida de estas funciones proveen un codigo de error que sirve como salida del programa. Los mensajes de versión y ayuda se imprimen por salida estándar (stdout) y el programa finaliza devolviendo 0 (cero) al sistema. Los mensajes de error se imprimen por la salida de errores (stderr) y el programa finaliza devolviendo 1 (uno) al sistema.

2.2.1. Errores posibles

1. El procesamiento de la entrada estándar causó el agotamiento del heap.
2. La invocación del programa es incorrecta.
3. Alguno de los archivos es inexistente.

4. Completar con mas posibles errores...

Se contemplan otros errores gracias al uso de la variable externa errno. Cuando ocurre un error inesperado, el mismo es informado por stderr (con la funión perror()).

2.3. Desarrollo de actividades

1. Se instaló en un linux un repositorio de fuentes (SVN) para que al dividir las tareas del TP se pudiese hacer una unión de los cambios ingresados por cada uno de los integrantes más fácilmente.

2. Cada persona del grupo se comprometió a que sus cambios en el el código fuente y los cambios obtenidos del SVN que pudiesen haber subido los otros integrantes del grupo, sean compilados en diferentes sistemas operativos: Windows, Linux y el GXEmul, asegurando así la máxima portabilidad entre plataformas planteada en el enunciado.

3. Se estableció que todos los integrantes en mayor o menor medida, contribuyan en el desarrollo de todas las partes del código para que nadie quede en desconocimiento de lo que se hizo en cada sección. Si bien cada parte del código está comenazada por diferentes integrantes (parseo de los argumentos, lectura de los ficheros, etc), todos nos familiarizamos con cada una de estas partes y cumplimos la función de testers de lo hecho por otros integrantes.

4. Se propuso como meta paralela, hacer el programa lo mas reutilizable posible tratando de que los métodos desarrollados, sean los suficientemente modulares como para su posible reutilización en los TPs venideros.

5. Debido desconocimiento de LaTeX [**?**] para algunos integrantes del grupo, uno de los integrantes dió una breve introducción de como desarrollar este informe para que todos pudiésen modificarlo y agregar lo que considerase necesario. Para compilar el archivo del informe se usaron el compilador de LaTeX [**?**] propio que viene en el Linux, y en el caso que sea necesario compilar el informe en un Windows se uso el MIKTEX el cuál es una implementación del compilador de LaTeX [**?**] para dicha plataforma.

6. Para poder generar el código assembler a partir del código fuente, dentro de NETBSD se utilizó gcc con la siguiente opción:

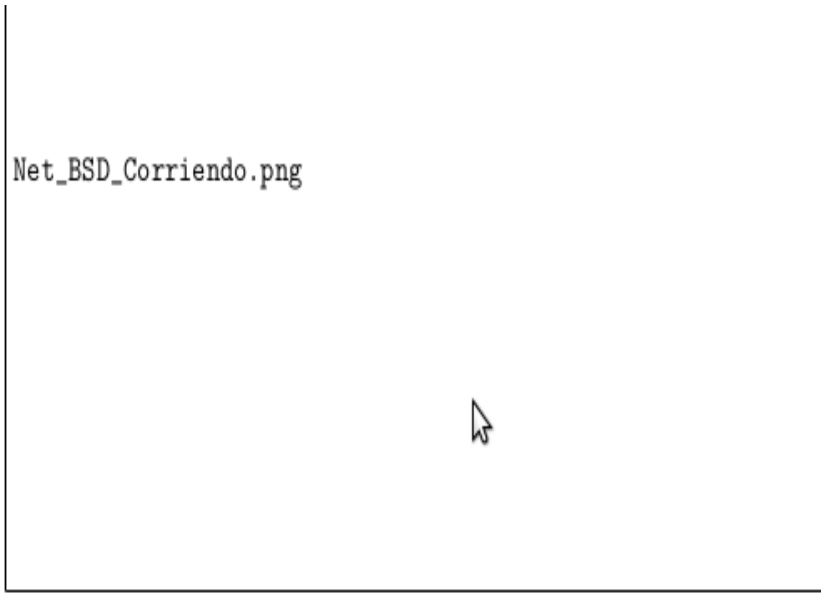
   gcc -S -Wall -O0 tp0.c -mrnames

7. Para crear el presente informe en formato PDF usando LaTeX [**?**] en Linux, ingresar los siguientes comandos:

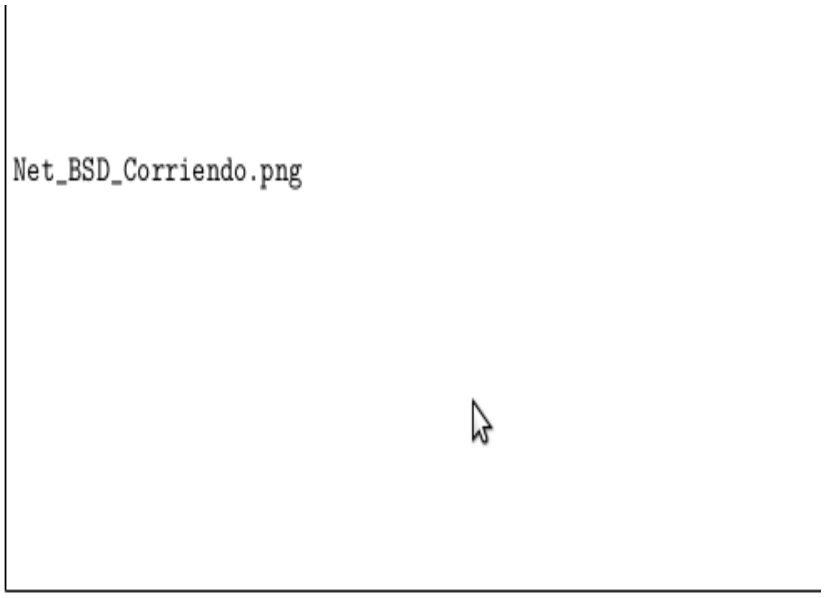
   $ pdflatex tp0.tex tp0.pdf

2.4. Screenshots

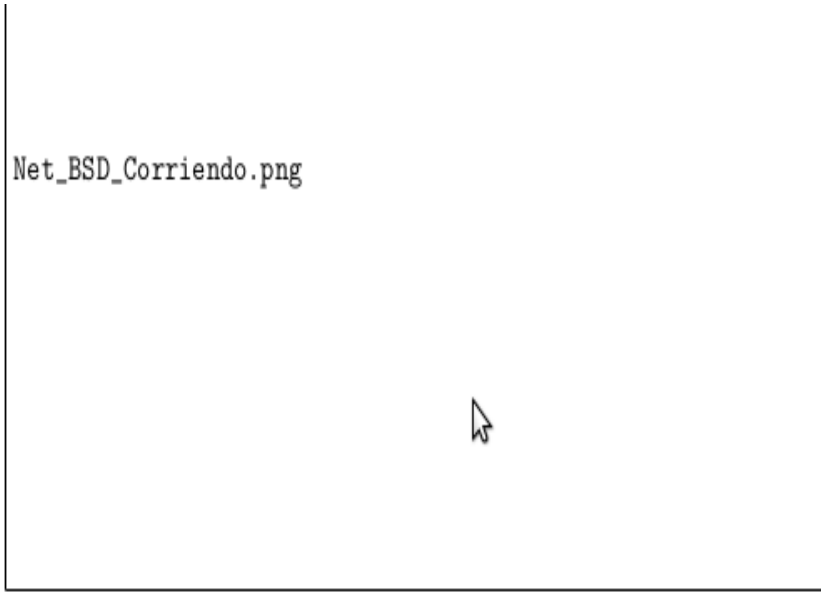
1. NetBSD corriendo



2. Túnel SSH
3. Linux a NetBSD

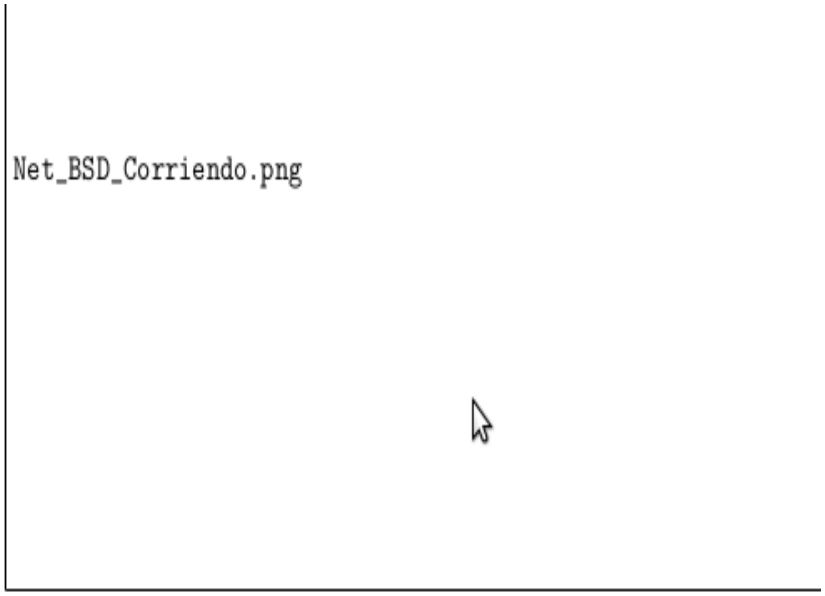


4. NetBsd a Linux

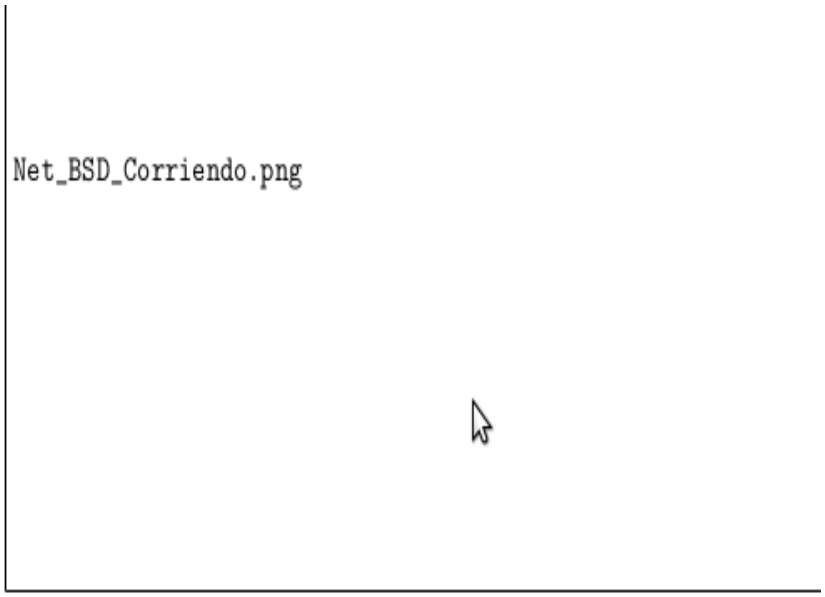


5. Programa compilado y corriendo en NetBSD

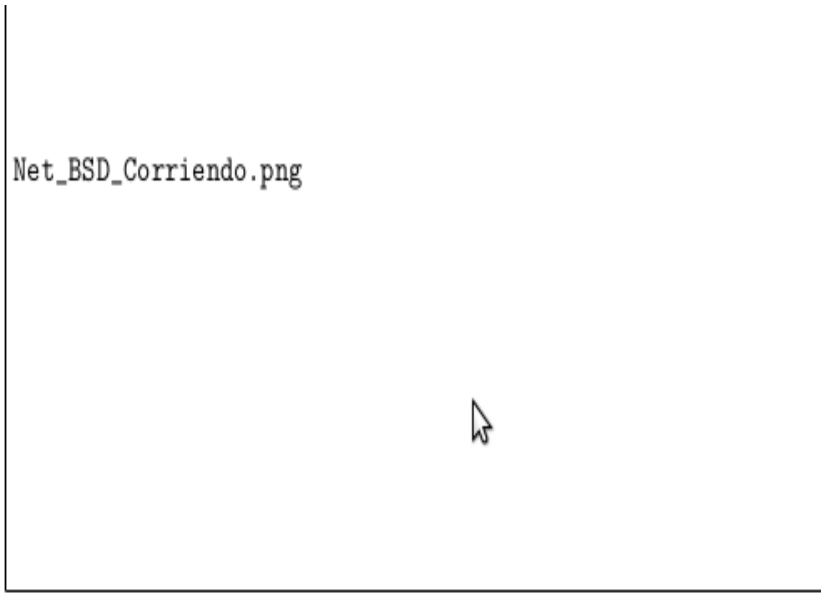
6. Mensaje de ayuda



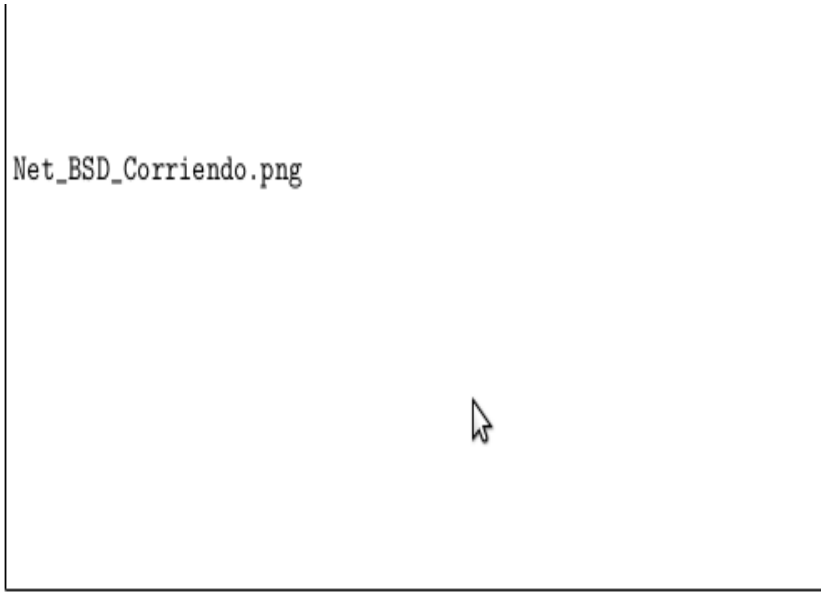
7. Mensaje de Versión



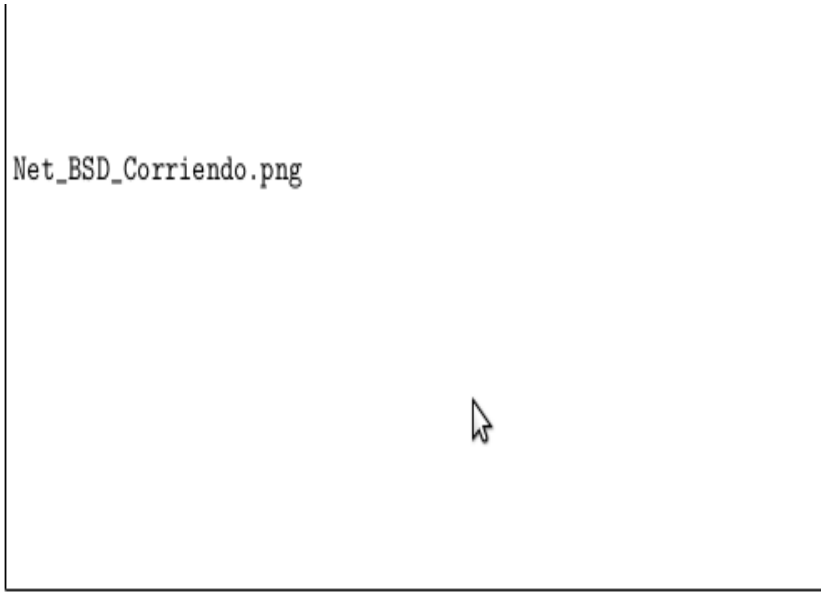
8. Pruebas
9. Archivo Alumnos.txt



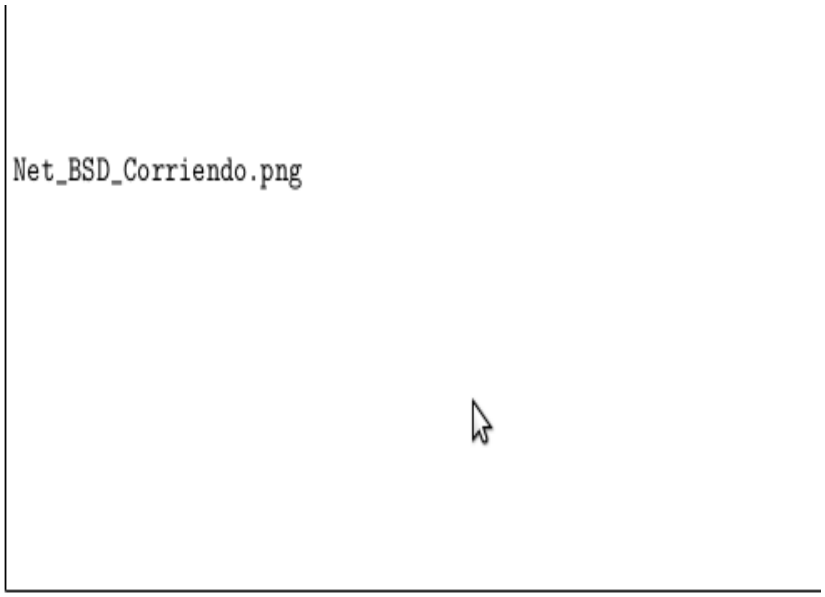
10. Varios Archivos



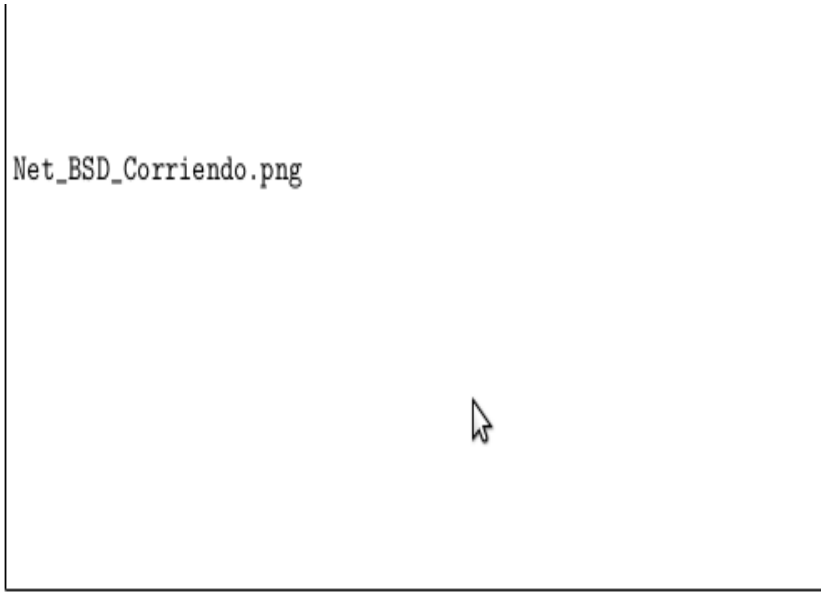
11. Stdin y Archivos



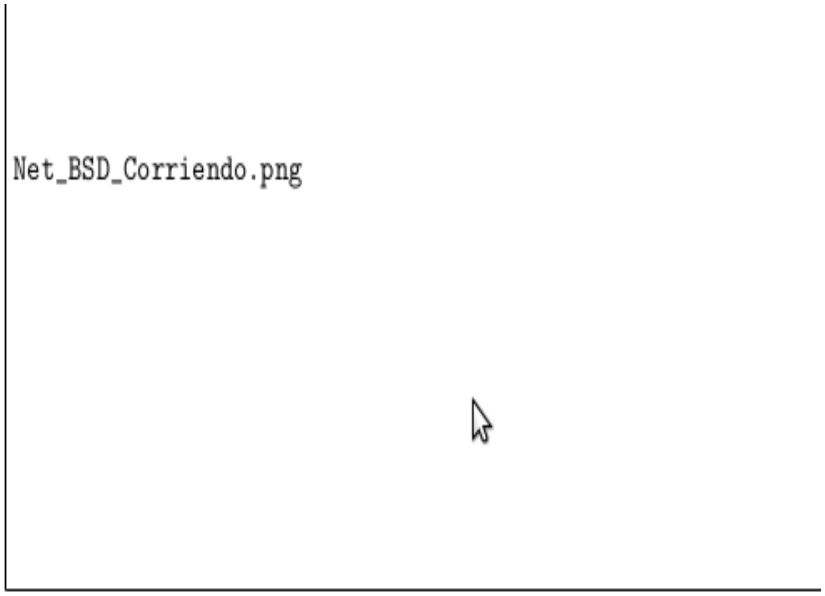
12. Solo Stdin



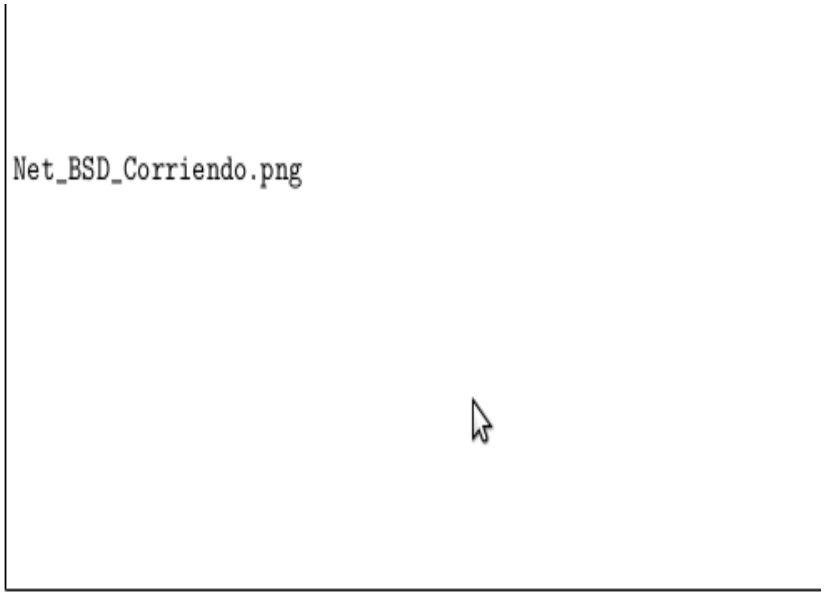
13. Errores
14. Archivo regular.



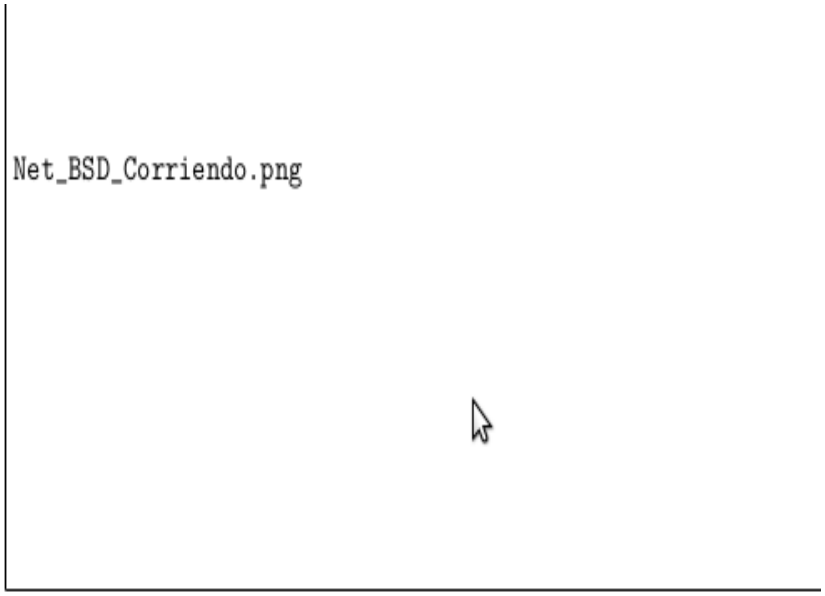
15. Archivo inexistente.



16. Mala Invocación.



17. Código assembler



3. Conclusiones

1. Si bien el lo solicitado por el programa no era excesivamente difícil, la realización completa del TP llevó cierta dificultad al tener que realizarlo en el contexto solicitado: alta portabilidad, desarrollo en C, e informe hecho en LaTeX [**?**].

2. En el primer caso la dificultad radicaba en tener configurado y funcionando el GXEmul dentro de un Linux, y lograr que en ambos casos el programa compile y corra sin problemas.

3. Debido a nuestro desconocimiento con LaTeX [**?**], tuvimos que invertir tiempo en encontrar forma de realizar el presente documento de la manera más correcta posible

4. En cuanto al trabajo grupal en si mismo, no hubo inconvenientes de ningún tipo ya que al ser el grupo relativamente chico y tener conocimiento del manejo del versionado de un proyecto ante cambios ingresado por los integrantes (por medio del SVN), la introducción de modificaciones y correcciones fué fluida.