

Herramientas de Ingeniería

Makefile

Se hace uso de archivos makefile para la implementación del proyecto, según Naranjo David:

“Si desean ejecutar o actualizar una tarea cuando se actualizan determinados archivos, la utilidad make puede venir muy bien. La utilidad make requiere de un archivo, makefile, que define un conjunto de tareas a ejecutar.”

Un archivo makefile básicamente se distingue en cuatro tipos básicos de declaraciones:

Comentarios

Variables.

Reglas explícitas.

Reglas implícitas.

Las Reglas explícitas le indican a make que archivos dependen de otros archivos, así como los comandos requeridos para compilar un archivo en particular.

Mientras que las implícitas son similares que las explícitas, pero con la diferencia indican los comandos a ejecutar, sino que make utiliza las extensiones de los archivos para determinar que comandos ejecutar.

En este ejemplo no vamos a utilizar estas últimas dos, solamente vamos a empezar por la impresión del clásico «Hola Mundo» en el terminal.

Para ello vamos a crear un directorio vacío con el nombre que deseen.

En este caso vamos a llamarlo “ejemplo-make” y dentro de este vamos a crear un un archivo makefile con el siguiente contenido:

```
#este es un comentario, todo lo que esté dentro de esta línea simplemente es  
ignorado  
esta_es_una_variable:  
echo "Hola Mundo"
```

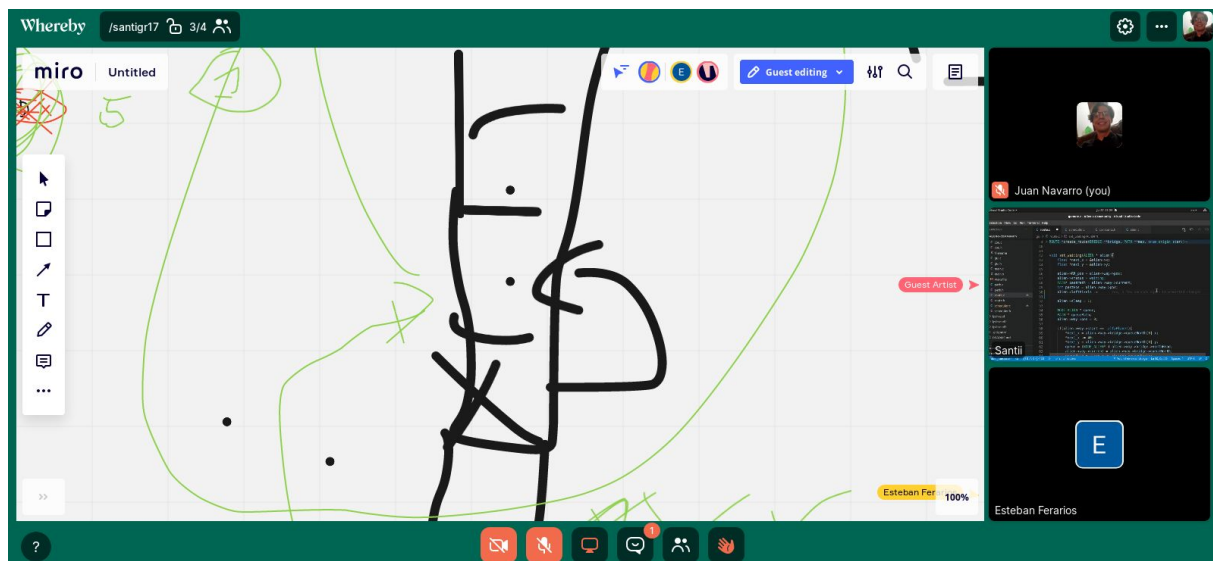
Ahora ejecuta el archivo escribiendo make dentro del directorio, la salida será:

```
make  
echo "Hola Mundo"  
Hola Mundo
```

Reuniones como herramientas de dibujo:

La reuniones mediante el sitio web WhereBy permitieron que el equipo pueda realizar observaciones, explicar abordajes de algoritmos y compartir recursos por tiempo ilimitado para usuario de 4 personas o menos.

El valor agregado fue la pizarra y complementos para que todos los usuarios puedan editar el contenido y participar en la construcción de los algoritmos



Git Flow, Pull Request e Issues

Mediante la herramienta de los issues se pudo identificar diversas de las tareas claves que cada miembro del grupo se estaban encargando. Con el fin de tener la seguridad de los acuerdo que se estaban tomando. Esta herramienta está integrada por defecto en la plataforma github.



Imagen de un issue abierto y 7 issues cerrados.

La estrategia de git flow nos permitió ir desarrollando una serie de cambios incrementales al proyecto en el branch master, mediante la creación de features los

miembros del equipo iban desarrollando mayores funcionalidades con el fin de ir establecido de forma segura cambios al proyecto principal.

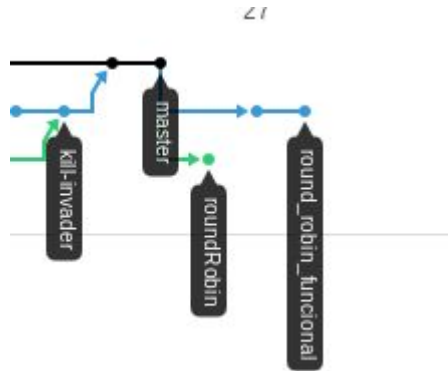


Imagen de integración de cambios de los diversos branches.



Tarjeta de pull en la plataforma de GitHub

Cada pull request permitía establecer de forma segura la solicitud de agregar nuevos código, permitiendo agregar cambios y que el nuevo código sea identificado para respectiva revisión.

GCC

Según Victor Gonzalez: “GCC es un compilador integrado del proyecto GNU para C, C++, Objective C y Fortran; es capaz de recibir un programa fuente en cualquiera de estos lenguajes y generar un programa ejecutable binario en el lenguaje de la máquina donde ha de correr.

La sigla GCC significa "GNU Compiler Collection". Originalmente significaba "GNU C Compiler"; todavía se usa GCC para designar una compilación en C. G++ refiere a una compilación en C++.”

Sintaxis.

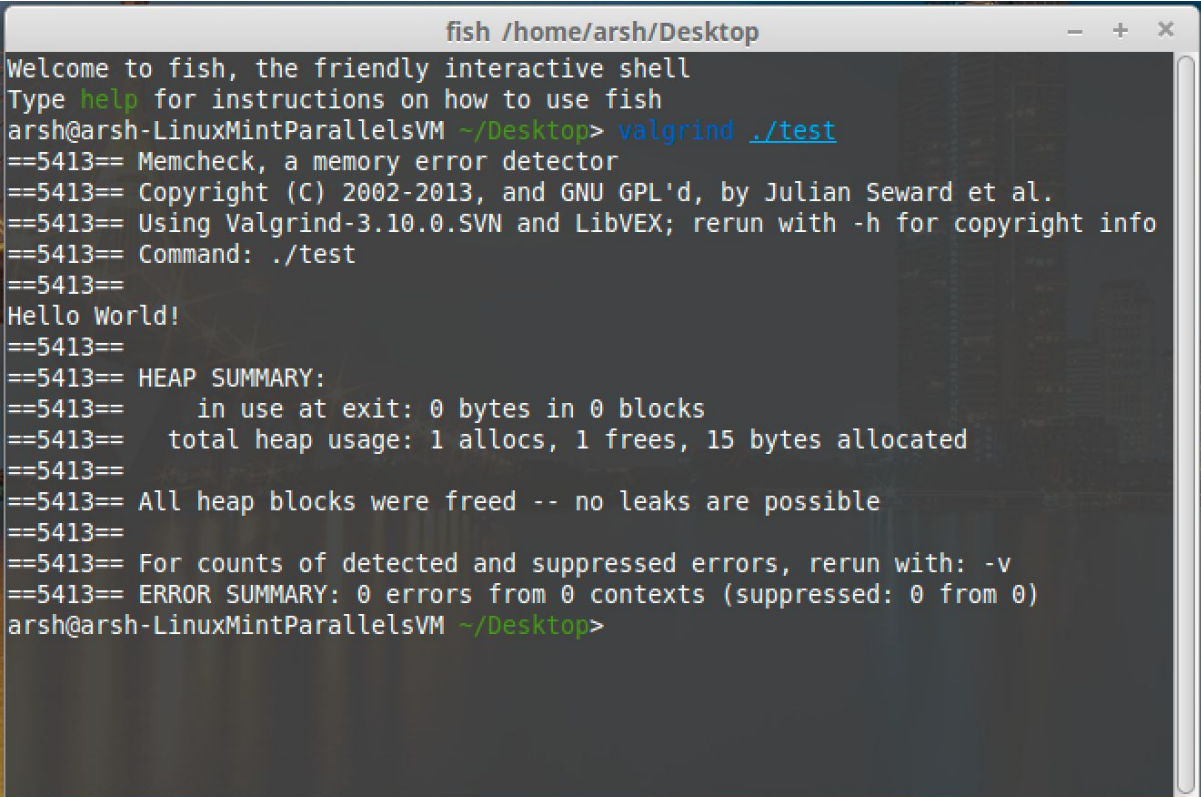
```
gcc [ opción | archivo ] ...
g++ [ opción | archivo ] ...
```

Las opciones van precedidas de un guión, como es habitual en UNIX, pero las opciones en sí pueden tener varias letras; no pueden agruparse varias opciones tras un mismo guión. Algunas opciones requieren después un nombre de archivo o directorio, otras no. Finalmente, pueden darse varios nombres de archivo a incluir en el proceso de compilación.

Valgrind

TOMADO DE: [Página oficial](#): El conjunto de herramientas Valgrind proporciona una serie de herramientas de depuración y creación de perfiles que lo ayudan a hacer que sus programas sean más rápidos y correctos. La más popular de estas herramientas se llama Memcheck. Puede detectar muchos errores relacionados con la memoria que son comunes en los programas C y C ++ y que pueden provocar bloqueos y comportamientos impredecibles.

Esta herramienta fue valiosa en las pruebas para detectar la creación de los hilos y diversos errores asociados a posibles accesos a datos cuando su memoria se liberó.

A screenshot of a terminal window titled 'fish /home/arsh/Desktop'. The terminal shows the output of running 'valgrind ./test'. The output includes a welcome message for fish, the command being run, copyright information for Valgrind, a 'Hello World!' message, a 'HEAP SUMMARY' section showing 0 bytes in use at exit and 15 bytes allocated, and an 'ERROR SUMMARY' section showing 0 errors. The prompt is 'arsh@arsh-LinuxMintParallelsVM ~/Desktop>'.

```
fish /home/arsh/Desktop
Welcome to fish, the friendly interactive shell
Type help for instructions on how to use fish
arsh@arsh-LinuxMintParallelsVM ~/Desktop> valgrind ./test
==5413== Memcheck, a memory error detector
==5413== Copyright (C) 2002-2013, and GNU GPL'd, by Julian Seward et al.
==5413== Using Valgrind-3.10.0.SVN and LibVEX; rerun with -h for copyright info
==5413== Command: ./test
==5413==
Hello World!
==5413==
==5413== HEAP SUMMARY:
==5413==   in use at exit: 0 bytes in 0 blocks
==5413== total heap usage: 1 allocs, 1 frees, 15 bytes allocated
==5413==
==5413== All heap blocks were freed -- no leaks are possible
==5413==
==5413== For counts of detected and suppressed errors, rerun with: -v
==5413== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
arsh@arsh-LinuxMintParallelsVM ~/Desktop>
```

GDB

TOMADO de la [página oficial](#):

GDB puede hacer cuatro tipos principales de cosas (además de otras cosas en apoyo de estas) para ayudarlo a detectar errores en el acto:

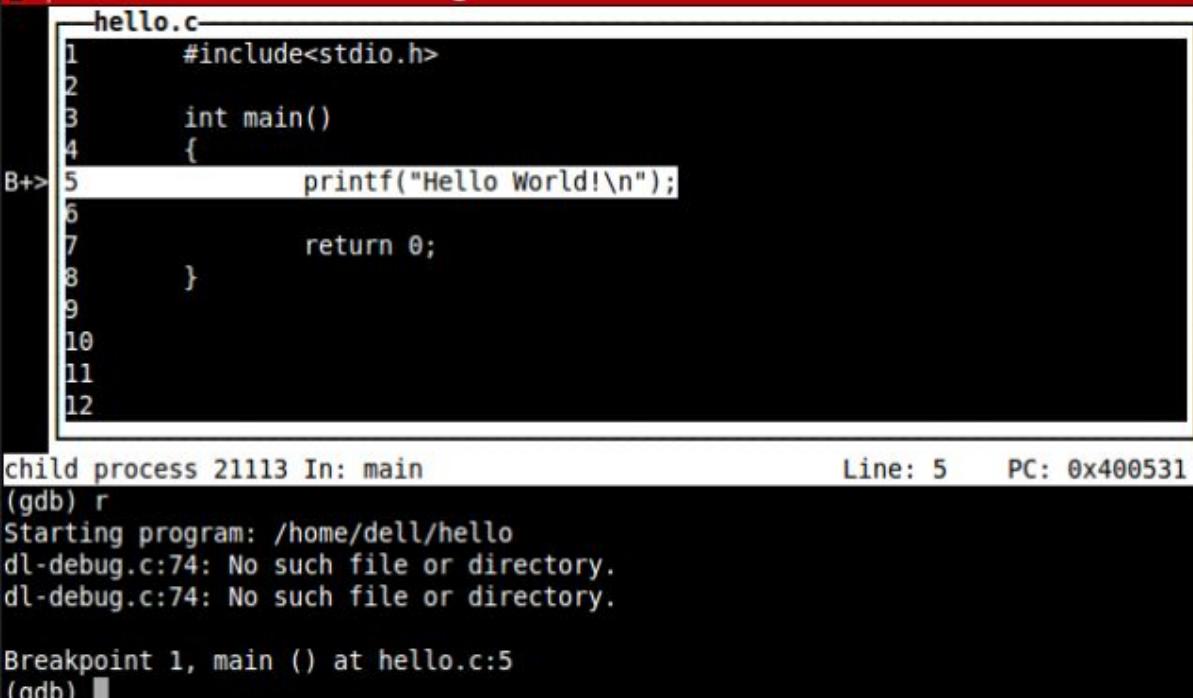
- Inicie su programa, especificando cualquier cosa que pueda afectar su comportamiento.

- Haga que su programa se detenga en condiciones específicas.

- Examine lo que sucedió cuando su programa se detuvo.

Cambie las cosas en su programa, para que pueda experimentar con la corrección de los efectos de un error y continuar aprendiendo sobre otro.

De lo anterior, fue valioso específicamente en la validación de errores para la manipulación del movimiento de los aliens, esto implica una ventaja pues se podía observar las diversas variables e información previo a la deliberación de los casos del desplazamiento de un alien.



```
hello.c
1  #include<stdio.h>
2
3  int main()
4  {
5      printf("Hello World!\n");
6
7      return 0;
8  }
9
10
11
12

child process 21113 In: main          Line: 5    PC: 0x400531
(gdb) r
Starting program: /home/dell/hello
dl-debug.c:74: No such file or directory.
dl-debug.c:74: No such file or directory.

Breakpoint 1, main () at hello.c:5
(gdb) █
```

Aprendizaje Continuo

Con cada trabajo grupal, se mejora la interacción con los otros compañeros de trabajo, además de como interpretar mejores soluciones a los trabajos propuestos en este caso las tareas 1 y 2 adicionalmente se aprende a vincular los nuevos conceptos que se aprenden hasta el momento, para lograr cada vez soluciones más eficientes, así como el uso de nuevas herramientas para lograr dichas soluciones. El efecto de la investigación previa sobre los procesos y en especial el taller 3 permitieron dominar con mayor detenimiento los hilos.

Referencias

D. Naranjo, "¿Qué es un archivo Makefile y cómo funciona dentro de Linux?", *Desde Linux*, 2020. [Online]. Available:

<https://blog.desdelinux.net/que-es-un-archivo-makefile-y-como-funciona-dentro-de-linux/>. [Accessed: 20- May- 2020].

V. González, "El compilador GCC - Tutorial", *lie.fing.edu.uy*. [Online]. Available: <https://iie.fing.edu.uy/~vagonbar/gcc-make/gcc.htm>. [Accessed: 20- May- 2020].

Fernández, G., 2017. Crea Rápidamente Servicios Con Systemd Para Iniciar Demonios O Ejecutar Scripts - Poesía Binaria. [online] Poesía Binaria. Available at: <<https://poesiabinaria.net/2017/12/crea-rapidamente-servicios-systemd/>> [Accessed 18 Abril 2020].

