

Ya habéis visto como compilar un programa fuente (hola_mundo.c) tanto en un ejecutable, como dejarlo en un paso intermedio, el programa objeto y de ahí, ver el código máquina y el ensamblador.

Ahora vamos a ver cómo trabaja un depurador para programas ejecutables. Para ello toma el fichero

puntoruptura.c adjunto a esta tarea y llevatelo a local (al mismo sitio de la práctica de compilación con gcc).

El programa que permite depurar programas escritos en C es gdb, vamos a instalarlo:

- Windows
- como en la práctica de gcc, ejecutamos el instalador de MinGW e instalamos la extensión mingw32-gdb-bin.
- Linux
- al instalar gcc, posiblemente se instaló gdb, prueba a ejecutarlo. Si no instalalo.

En este punto ya tenemos gdb instalado.

1. Ahora vamos a usar gdb para ver cómo se hacen puntos de ruptura en la ejecución de un programa (breakpoints) y entender un poco como se depuran lenguajes compilados, en este caso C. Para poder llevar un ejecutable al depurador hay que compilarlo añadiendo el parámetro "-g". Compilalo como lo vimos en la tarea de gcc. Que el fichero compilado se llame "puntoruptura".

2. Linux

1. Ahora ejecuta el depurador mediante "gdb -tui" te aparecerá la pantalla dividida en dos, arriba aparecerá el código y abajo una línea para introducir comandos que comienza por (gdb)

3. Windows

1. En Windows simplemente ejecuta gdb.

4. Cargáis el ejecutable en el depurador mediante "file puntoruptura". En la parte de arriba

podéis ver el código cargado (en Linux únicamente, mira cómo se ve en la pantalla de algún compañero que tenga Linux)

5. Ahora ponemos un punto de ruptura mediante "break 14", marcará en la parte de arriba el breakpoint mediante B+> (de nuevo solo en Linux)

6. Comenzamos el programa mediante "run" y la ejecución se para en el primer printf (da igual el número de línea en realidad, el depurador va a la línea de código más cercana a la línea que has marcado).

7. Ahora podemos ver el valor de las variables a y b poniendo print a y print b.

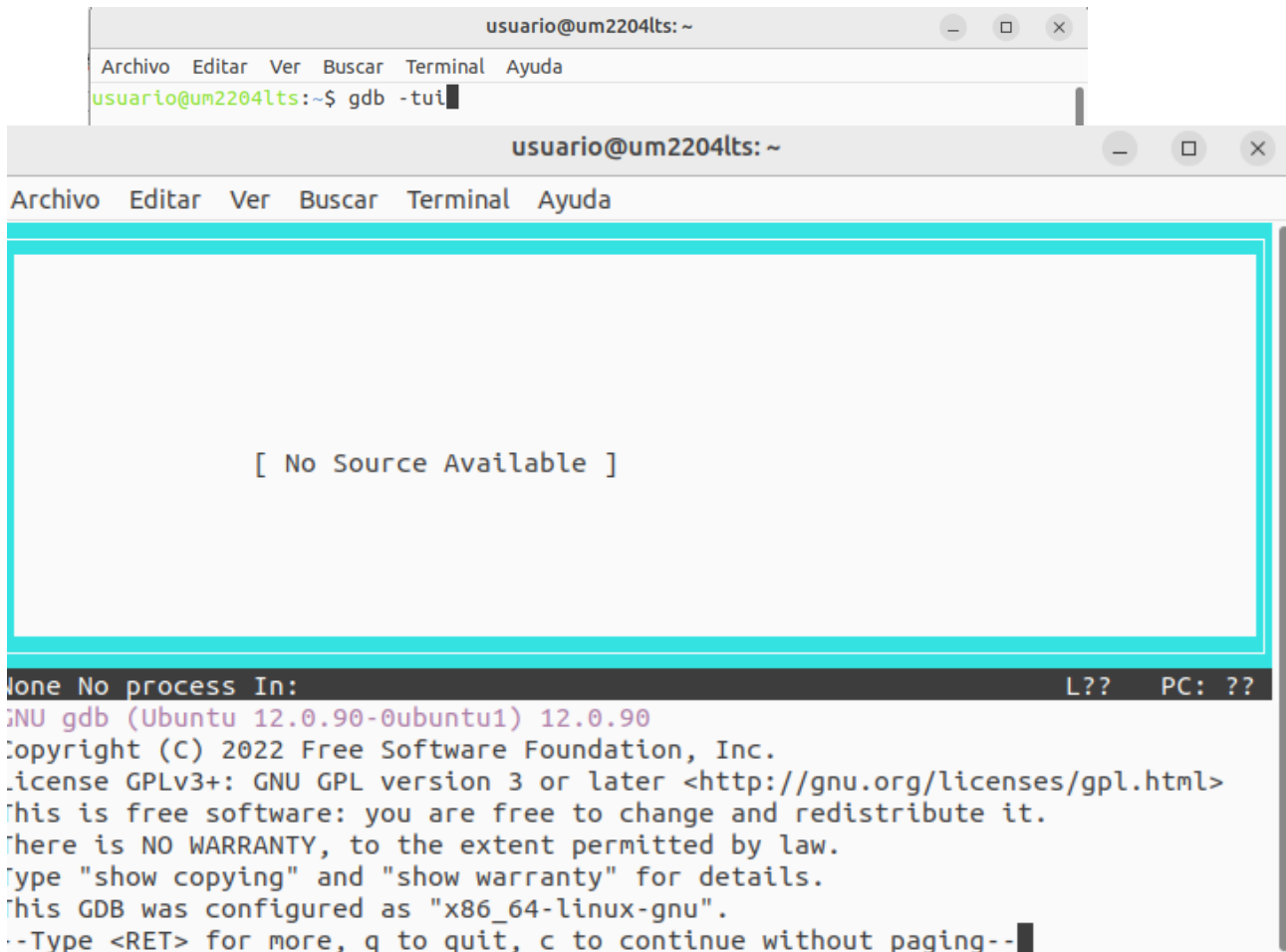
8. Incluso podemos modificarlas en tiempo de ejecución mediante "set var b = 100000000".

9. Podemos seguir la ejecución del programa tecleando n ("next") hasta la finalización.

10. Haced pantallazos del proceso, usad la plantilla, y convertirla en pdf y subid al repo de GitHub que aparece en la tarea.

11. Además en la misma plantilla al final contestad a las preguntas siguientes:

1. ¿Qué es un depurador, y cuál es su función.?
2. ¿Cómo crees que es la depuración en un lenguaje interpretado?
3. ¿Cómo crees que es la depuración en un lenguaje con una maquina virtual?



```
usuario@um2204lts: ~
Archivo Editar Ver Buscar Terminal Ayuda
usuario@um2204lts:~$ gdb -tui

[ No Source Available ]

Done No process In: L?? PC: ??
GNU gdb (Ubuntu 12.0.90-0ubuntu1) 12.0.90
Copyright (C) 2022 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
--Type <RET> for more, q to quit, c to continue without paging--
```

1. ¿Qué es un depurador, y cuál es su función.?

Un depurador (en inglés: debugger) es un **programa que se utiliza para detectar e identificar los errores en un software** (el programa "objetivo") y, por lo tanto, los desarrolladores pueden encontrar los fallos en el programa de forma más fácil, facilitando el proceso de corrección.

2. ¿Cómo crees que es la depuración en un lenguaje interpretado?

Un **lenguaje interpretado** es un **lenguaje** de programación para el que la mayoría de sus implementaciones ejecuta las instrucciones directamente, sin una previa compilación del programa a instrucciones en **lenguaje** máquina.

3. ¿Cómo crees que es la depuración en un lenguaje con una maquina virtual?

Puede ahorrar tiempo y dinero mediante el emulador de Proceso de Azure para depurar el servicio en la nube en un equipo local. La depuración de un servicio localmente antes de su implementación puede mejorar la fiabilidad y el rendimiento sin pagar por tiempo de proceso. No obstante, pueden producirse algunos errores solo al ejecutar un servicio en la nube en el propio Azure. Puede depurar estos errores si habilita la depuración remota al publicar el servicio y luego adjuntar el depurador a una instancia de rol.