

Entorn de treball

Miquel Albert
Gerard Enrique

Índex

Introducció	3
1. Instal·lació de les eines	5
1.1. Instal·lació de VirtualBox	5
1.2. Instal·lació del VirtualBox Extension Pack	6
1.3. Afegir un usuari al grup de VirtualBox (Linux)	6
1.4. Instal·lació d'una màquina virtual	6
1.5. Instal·lar les Guest Additions de VirtualBox	19
1.6. Carpetes compartides	22
1.7. Porta-retalls compartit i arrossegar fitxers	24
1.8. Accedir a un dispositiu USB	25
1.9. Instal·lació de les eines	25
2. Utilització de les eines	28
2.1 Editor de text: geany	28
2.2. Assemblador: yasm	28
2.3. Enllaçador: gcc (ld)	29
2.4. Compilador de C: gcc	30
2.5. Depurador : kdbg (gdb)	30
2.6. Execució	31
3. Procés de desenvolupament en assemblador	32
3.1. Edició del codi font	32
3.2. Assemblatge del codi font	33
3.3. Enllaçament del codi objecte i generació de l'executable	34
3.4. Execució del programa	34
3.5. Depuració del programa amb KDbg (gdb)	34
4. Procés de desenvolupament en C i assemblador	40
4.1. Edició del codi font assemblador	40
4.2. Assemblatge del codi font assemblador	41
4.3. Edició del codi font C	41
4.4. Compilació del codi font C, assemblatge amb del codi objecte assemblador i generació de l'executable	41
4.5. Execució del programa	41
4.6. Depuració del programa amb KDbg	42

Introducció

L'entorn de treball que utilitzarem per a desenvolupar els problemes i les pràctiques de programació serà un PC basat en processadors x86-64 (Intel64 o AMD64) sobre el qual s'executarà un sistema operatiu Linux de 64 bits, la versió Linux que proposem és Linux Mint de 64 bits (basada en Ubuntu), però es poden utilitzar altres versions de Linux de 64 bits.

Per a la instal·lació de Linux Mint, com s'explica més endavant, haureu de descarregar un fitxer amb una imatge ISO des del lloc web de Linux Mint. El fitxer ISO permet fer una instal·lació del sistema en una màquina virtual, de VirtualBox per exemple, i també permet fer la instal·lació de forma nativa.

Els llenguatges de programació que utilitzarem per a escriure el codi font dels problemes i de les pràctiques de programació de l'assignatura seran el llenguatge C per dissenyar el programa principal i les operacions d'E/S i el llenguatge assemblador x86-64 per implementar funcions concretes i veure com treballa aquesta arquitectura a baix nivell.

Eines proposades

Editor: *geany*

Assemblador: *yasm*

Enllaçador: *gcc* (*internament crida a l'enllaçador ld*)

Compilador de C: *gcc*

Entorn de depuració: *kdbg* que utilitza *gdb*

Més endavant en aquest document s'explica com instal·lar i utilitzar aquestes eines.

Procés de desenvolupament d'un programa escrit en llenguatge assemblador:

1. Edició del codi font assemblador (*geany*).
2. Assemblatge del codi font assemblador i generació del codi objecte (*yasm*).
3. Enllaçament del codi objecte i generació del codi executable (*gcc*).
4. Depuració del codi executable per a la correcció d'errors (*kdbg*).
5. Execució del programa.

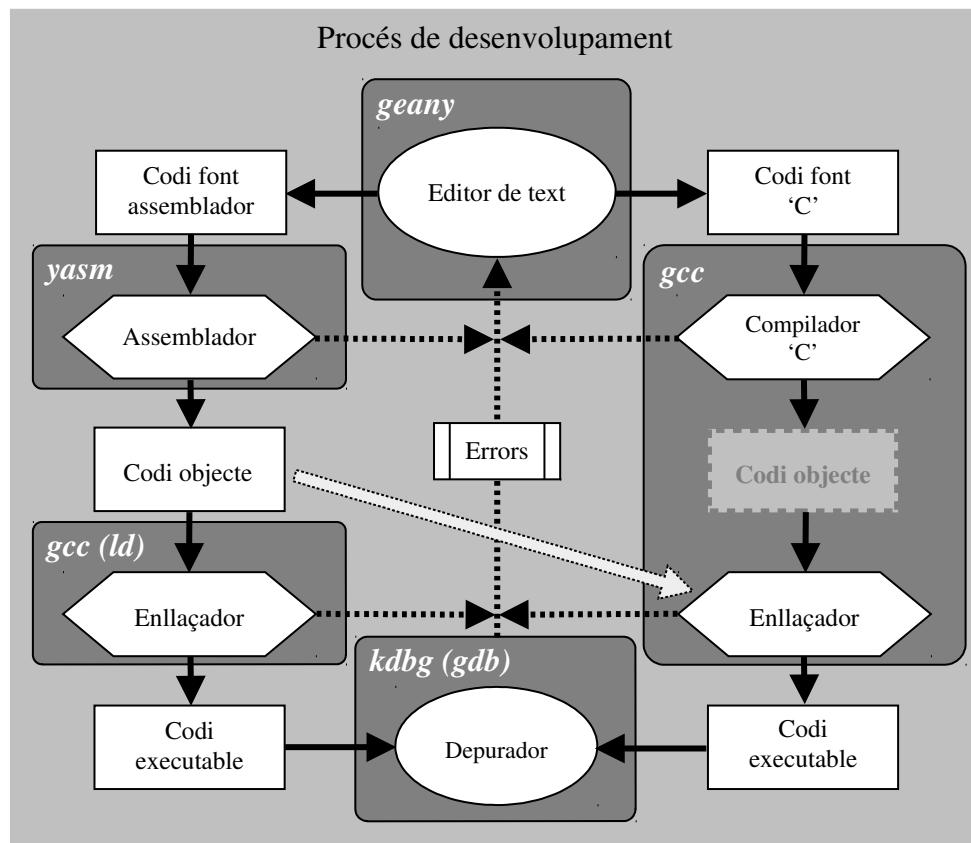
Procés de desenvolupament d'un programa escrit en llenguatge C:

1. Edició del codi font C (*geany*).
2. Compilació i enllaçament del codi font C i generació del codi executable (*gcc*).
3. Execució del programa.

Procés de desenvolupament d'un programa escrit en llenguatge C que utilitza subrutines fetes en assemblador:

1. Edició del codi font assemblador, incloure la definició de les subrutines necessàries com globals (*geany*).
2. Assemblatge del codi font assemblador i generació del codi objecte (*yasm*).
3. Edició del codi font C, incloses les definicions i les crides a les subrutines en assemblador (*geany*).
4. Compilació i enllaçament del codi font C amb el codi objecte generat de l'assemblador i generació del codi executable (*gcc*).
5. Depuració del codi executable per a la correcció d'errors (*kdbg*).
6. Execució del programa.

Quan en un d'aquests processos es detecten errors, cal tornar a l'inici del procés, modificar el codi font per a corregir els errors i repetir el procés cíclicament fins a obtenir un programa executable lliure d'errors i que tingui la funcionalitat desitjada.



1. Instal·lació de les eines

En aquest apartat s'explica com instal·lar i configurar les eines d'entorn de treball.

1.1. Instal·lació de VirtualBox

Primer de tot cal que comproveu si el vostre ordinador permet executar una màquina virtual amb un sistema operatiu de 64 bits.

Per a fer la comprovació, en Windows, podeu executar el programa “securable” que trobareu en el següent enllaç web:
<http://www.grc.com/files/securable.exe>

A l'executar aquesta eina apareixen dues informacions importants, si el processador és de 64 bits i si té suport per a virtualització, les dues característiques són necessàries per poder executar una màquina virtual amb un sistema operatiu de 64 bits dins de VirtualBox.

En Linux, podeu comprovar si el vostre processador és de 64 bits i té suport per a virtualització, amb l'ordre *lscpu*. Obriu un terminal de Linux i executeu l'ordre:

```
$ lscpu
```

Si entre la informació que apareix, hi ha la informació següent:

```
Architecture:           x86_64
CPU op-mode(s):        32-bit, 64-bit
...
Virtualization:        VT-x
```

Significa que sí teniu suport per a virtualització. Si teniu un processador AMD, a la informació sobre virtualització apareixerà AMD-V.

Si el vostre ordinador té un processador de 64 bits però no té suport de virtualització per hardware, haureu d'instal·lar Linux de 64 bits en una partició del disc o en un disc extern o memòria USB.

Si no sabeu com fer-ho contacteu amb el consultor de la vostra aula que us podrà ajudar.

1.1.1. Obtenció i instal·lació del programari de VirtualBox

Es pot obtenir la darrera versió (5.1) del programari de VirtualBox a través de la pàgina web següent:

<https://www.virtualbox.org/wiki/Downloads>

Trobareu versions per a Windows, Mac (OS X) i Linux. En el cas de Windows i OS X hi ha una única versió, en el cas de Linux escolliu l'opció que s'adapti millor a la vostra distribució.

Si utilitzeu Windows 10 com a sistema operatiu en el vostre ordinador, assegureu-vos d'instal·lar la darrera versió de VirtualBox.

Un cop descarregat el programari executeu-lo i seguiu les indicacions del mateix.

1.1.2. Instal·lació mitjançant el gestor de paquets (Linux)

En el cas de tenir una distribució Linux com a sistema operatiu amfitrió basada en Debian o Ubuntu teniu l'opció d'afegir el dipòsit de paquets de VirtualBox a l'eina de gestió de paquets de Linux, per fer-ho seguiu els passos següents, des d'una terminal de linux:

Executeu l'ordre:

```
$ wget -q http://download.virtualbox.org/virtualbox/debian/oracle_vbox.asc -O- | sudo apt-key add -
```

Escriviu tota l'ordre en una sola línia.

Afegiu la línia següent en el fitxer /etc/apt/sources.list :

```
deb http://download.virtualbox.org/virtualbox/debian xenial contrib
```

En funció de la vostra distribució substituïu xenial per vivid, utopic, trusty, raring, quantal, precise, lucid, jessie, wheezy, o squeeze.

Per exemple, per a LinuxMint 18 i Ubuntu 16.04 LTS (xenial), es pot fer:

```
$ sudo sh -c 'echo "deb http://download.virtualbox.org/virtualbox/debian xenial contrib" >> /etc/apt/sources.list'
```

Escriviu tota l'ordre en una sola línia.

Per a instal·lar el programari execueu les ordres següents

```
$ sudo apt-get update  
$ sudo apt-get install virtualbox-5.1
```

1.1.3. Instal·lació en Windows 8, 8.1 i 10

Aquestes versions de Windows incorporen un gestor de màquines virtuals propi de Microsoft, Hyper-V, en cas de trobar-se activat VirtualBox no funcionarà correctament.

Per desactivar Hyper-V feu clic amb el botó dret del ratolí sobre la icona del menú de Windows  i escolliu l'opció de menú *Programes i característiques*, en el panell de l'esquerra escolliu *Activa o desactiva les característiques de Windows* desmarqueu l'opció Hyper-V i accepteu els canvis, cal reiniciar l'equip per activar els canvis.

1.2. Instal·lació del VirtualBox Extension Pack

Aquest paquet habilita certes extensions com el controlador USB. Descarregueu-vos el paquet des de la pàgina de descarregues:

<https://www.virtualbox.org/wiki/Downloads>

Es tracta d'una mateixa descarrega per a totes les plataformes (Windows, OS X i Linux)

Obriu el fitxer descarregat des de l'explorador d'arxius, directament es detectarà com un complement de VirtualBox i s'obrirà amb aquest programa. Seguiu les indicacions del mateix.

En el cas de Linux, us demanarà la contrasenya de l'usuari per a realitzar tasques privilegiades.

1.3. Afegeir un usuari al grup de VirtualBox (Linux)

En el cas de tenir una distribució Linux com sistema operatiu amfitrió cal afegir l'usuari amb el qual treballau al grup d'usuaris vboxusers per tenir accés als dispositius USB des de dins de les màquines virtuals creades amb VirtualBox. Per fer-ho execueu l'ordre següent:

```
$ sudo usermod -a -G vboxusers nom_usuari
```

A continuació cal sortir de la sessió i tornar a entrar per activar els canvis.

1.4. Instal·lació d'una màquina virtual

A continuació s'explica la creació d'una màquina virtual Linux per a l'assignatura utilitzant el programari de virtualització VirtualBox.

Per poder crear la màquina virtual, primer heu de descarregar el fitxer ISO de Linux Mint, que serà la distribució de Linux que farem servir.

Cal descarregar una imatge ISO de 64 bits a través de l'enllaç:

<http://www.linuxmint.com/download.php>

Linux Mint es distribueix en quatre edicions diferents, segons l'entorn d'escriptori: Cinnamon, MATE (basada en GNOME2), KDE i Xfce.

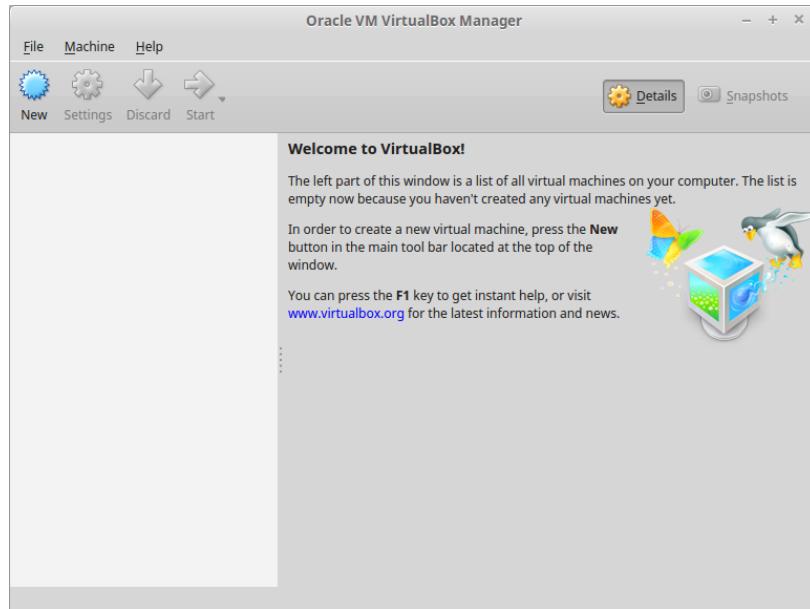
Podeu utilitzar qualssevol de les quatre edicions, però tingueu present el següent: les edicions Cinnamon i KDE necessiten més recursos hardware i són escriptoris pensats per a treballar amb acceleració de gràfics 3D, les edicions MATE i Xfce necessiten menys recursos i són més adequades per a treballar amb ordinadors amb menys recursos.

Descarregueu-vos la versió que vulgueu, sempre que sigui de 64 bits. Nosaltres recomanem utilitzar la versió MATE de 64 bits.

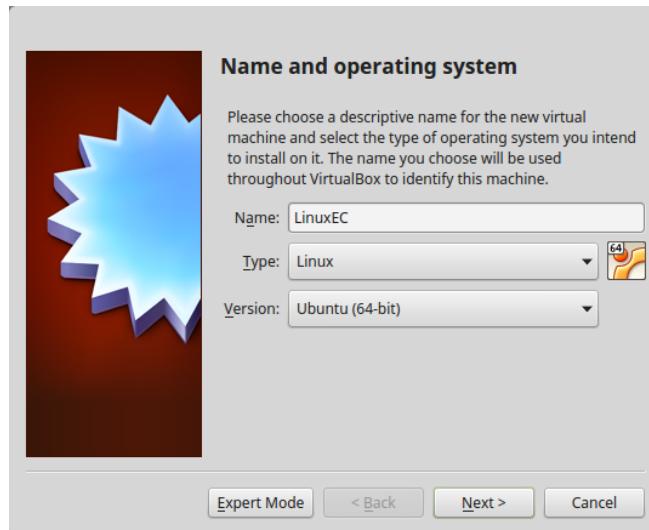
En aquest document se explicarà l'instal·lació a partir de l'edició MATE de 64 bits.

Un cop descarregat el fitxer ISO de la darrera versió, Linux Mint 18 MATE 64 bits, hauríeu de tenir el fitxer següent: *linuxmint-18-mate-64bit.iso*

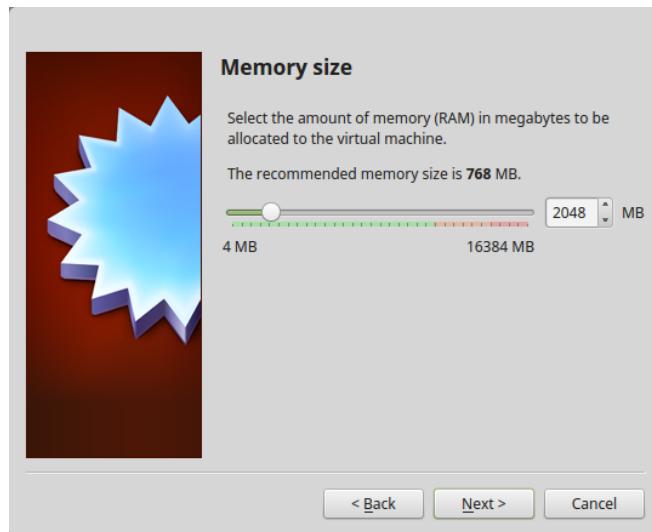
Obriu VirtualBox i escolliu l'opció *New*



En l'assistent que apareix doneu un nom a la màquina virtual i escolliu el tipus de sistema operatiu, Linux, i la versió, Ubuntu de 64 bits (ja que Linux Mint està basada en Ubuntu).



En la pantalla següent introduïu la quantitat de memòria, es recomana no sobrepassar el 50% de la memòria de l'ordinador, normalment amb 1024 MB és suficient, però podeu seleccionar una mida més gran, 2048 MB per exemple.



En la pantalla següent, es demana afegir un disc dur virtual, creeu un disc nou.

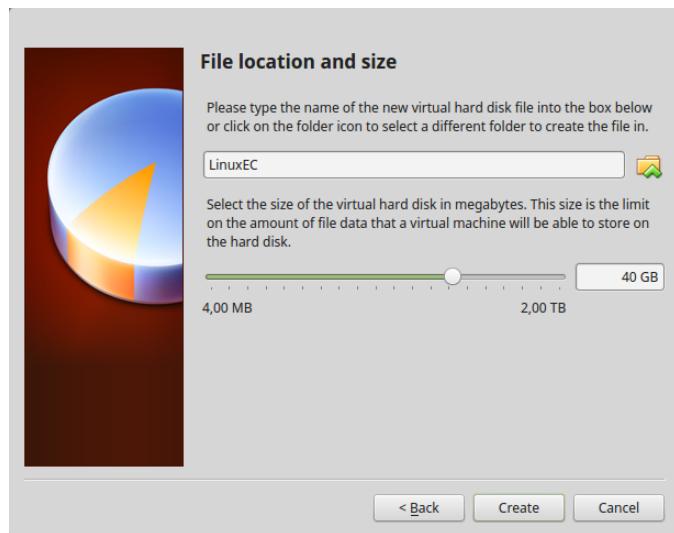


Escolliu un disc virtual de tipus VDI (VirtualBox Disk Image) ubicat de forma dinàmica.

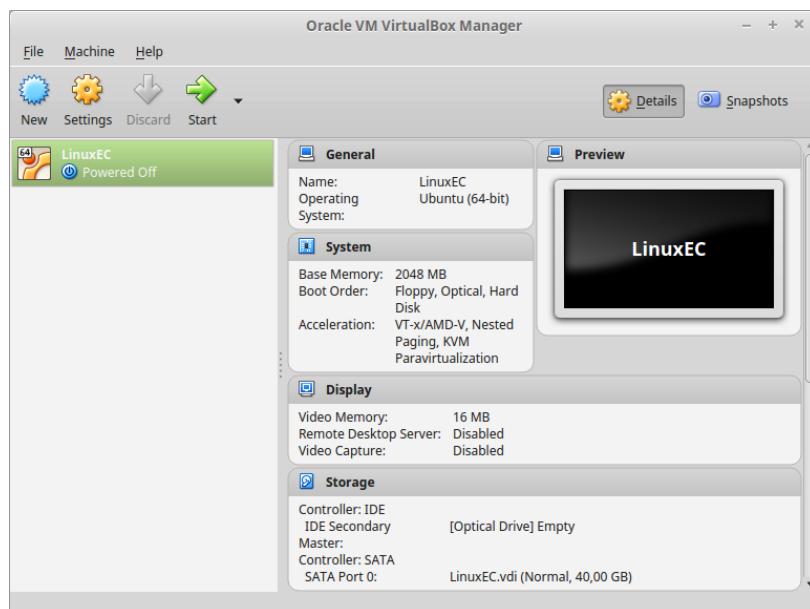


A continuació apareix una pantalla per triar la mida del disc.

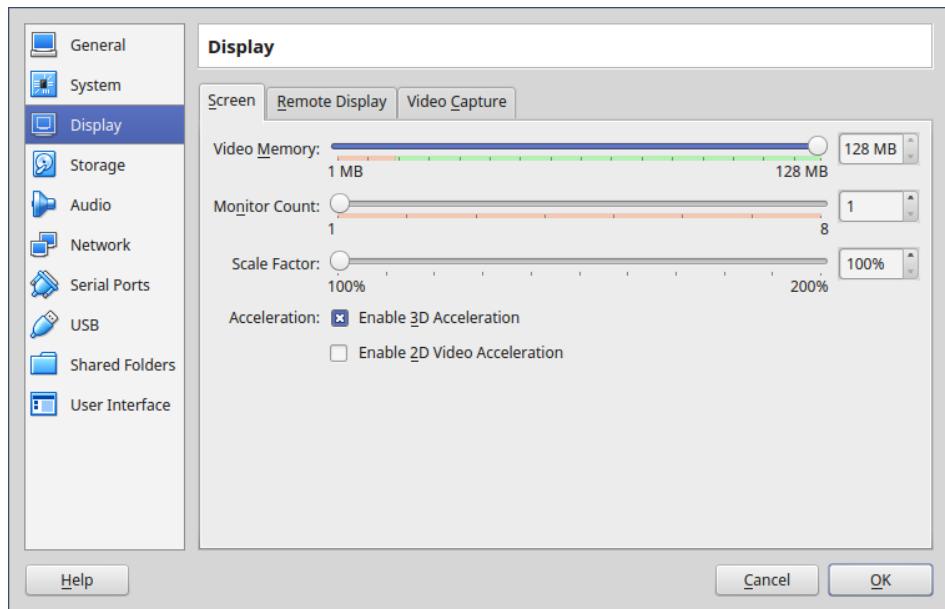
La instal·lació de LinuxMint necessita com uns 10 GB aproximadament, però si teniu espai suficient al disc podeu posar una mida superior, per exemple 40 GB. Aquesta serà la mida màxima, inicialment no s'occupa tota la mida, el fitxer del disc dur virtual va creixent a mesura del què és necessari.



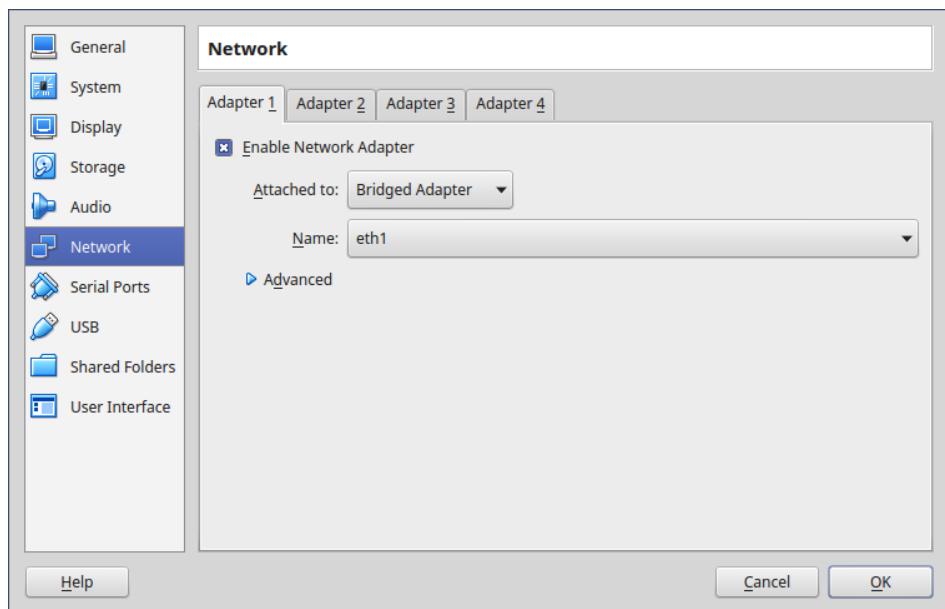
Accepteu el canvis i premeu el botó *Create*, es crearà el disc virtual i la màquina virtual, al finalitzar el procés us ha d'aparèixer una finestra com la següent:



Per evitar problemes amb la màquina virtual, us recomanem canviar la mida de la memòria de vídeo, posant la màxima possible, 128 MB, prement a sobre de l'apartat *Display*. També podeu millorar la presentació dels gràfics activant l'opció *Enable 3D Acceleration*.



També us recomanem canviar el tipus d'adaptador de xarxa, per fer-ho premeu sobre *Network*, en la finestra que apareix, premeu sobre el desplegable que hi ha al costat de l'etiqueta *Attached to* i seleccioneu *Bridged Adapter*.



Inicieu la nova màquina prement el botó *Start*.

Probablement apareixerà un missatge indicant que la màquina virtual capture el control del teclat i que el podeu alliberar prement la tecla *Ctrl Dreta*, accepteu el missatge, opcionalment podeu marcar que no us torni a avisar.

A continuació us apareixerà un assistent d'instal·lació en el qual podeu especificar el fitxer amb la imatge per a instal·lar el SO de la màquina virtual.

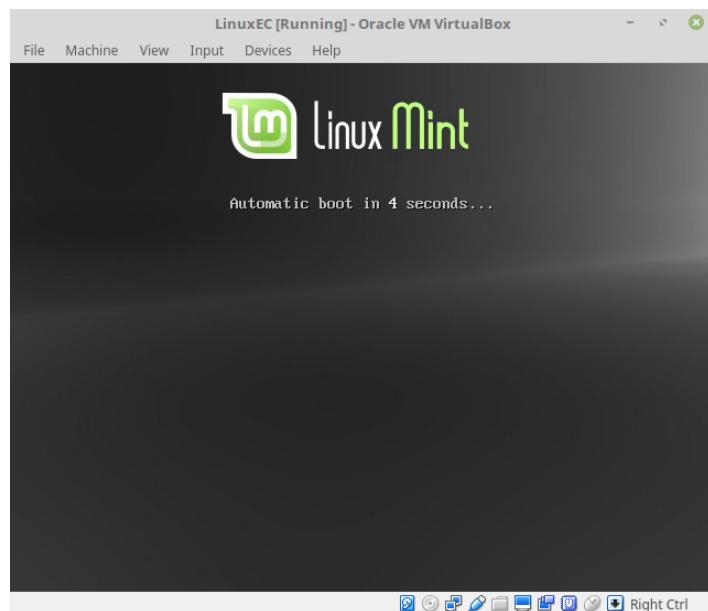
Premeu la icona amb forma de carpeta que apareix a la dreta, navegueu pel sistema de fitxers i escolliu el fitxer .iso: *linuxmint-18-mate-64bit.iso*

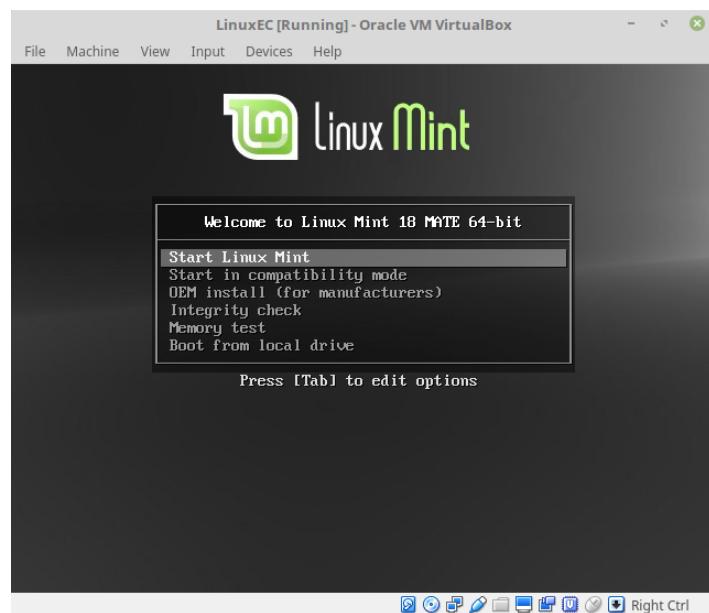


Premeu *Start* i s'iniciarà la instal·lació.

Si apareix un missatge indicant que es suporta la integració del punter o que la finestra està optimitzada per treballar amb 32 bits de color, accepteu el missatges, opcionalment podeu marcar que no us torni a avisar.

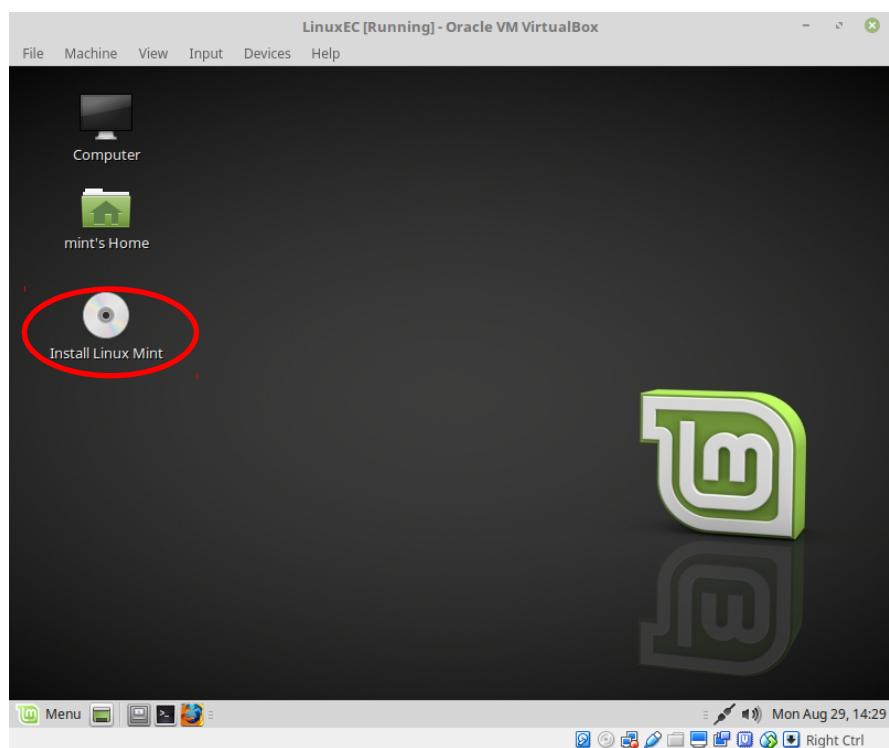
Apareix una pantalla amb un compte enrere, espereu a que finalitzi i arranqui automàticament o premeu Enter i en el menú que apareix, i escolliu Start Linux Mint.



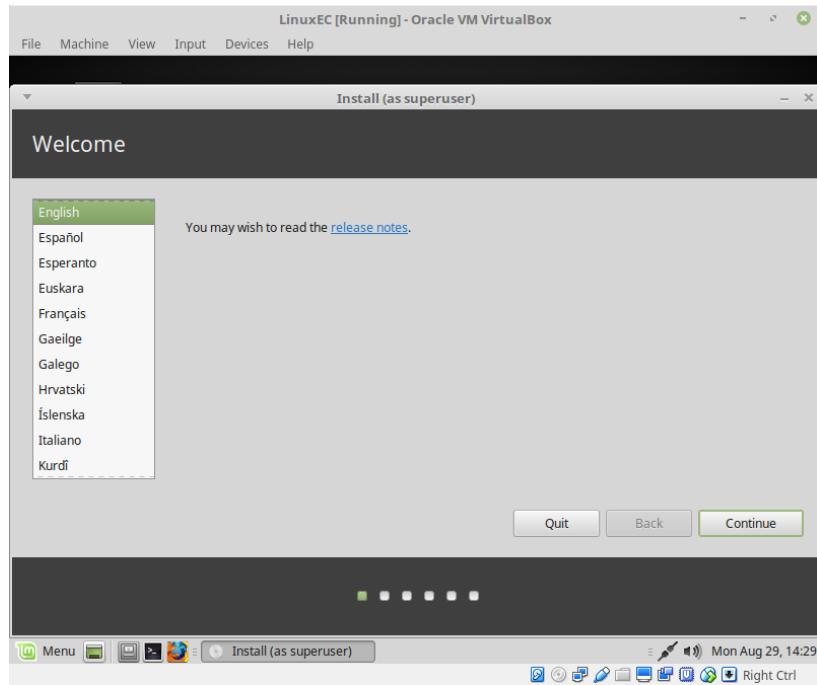


És possible que durant l'arrencada aparegui algun missatge d'advertència, no té cap importància, la instal·lació es podrà fer correctament.

Un cop arrancat el sistema inicie el programa d'instal·lació, fent doble clic amb el ratolí sobre l'ícone de l'escriptori Install Linux Mint.

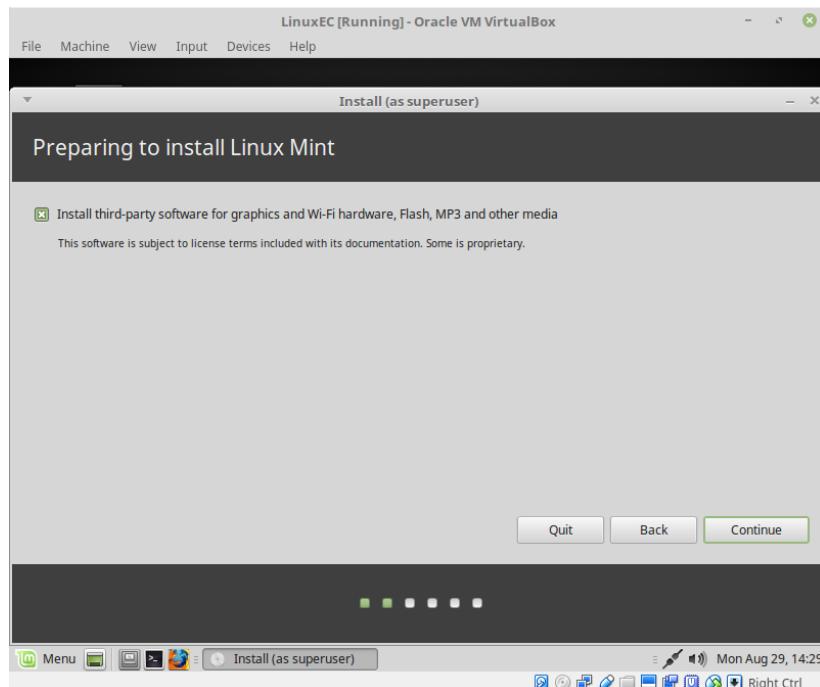


En la pantalla que apareix escolliu l'idioma:

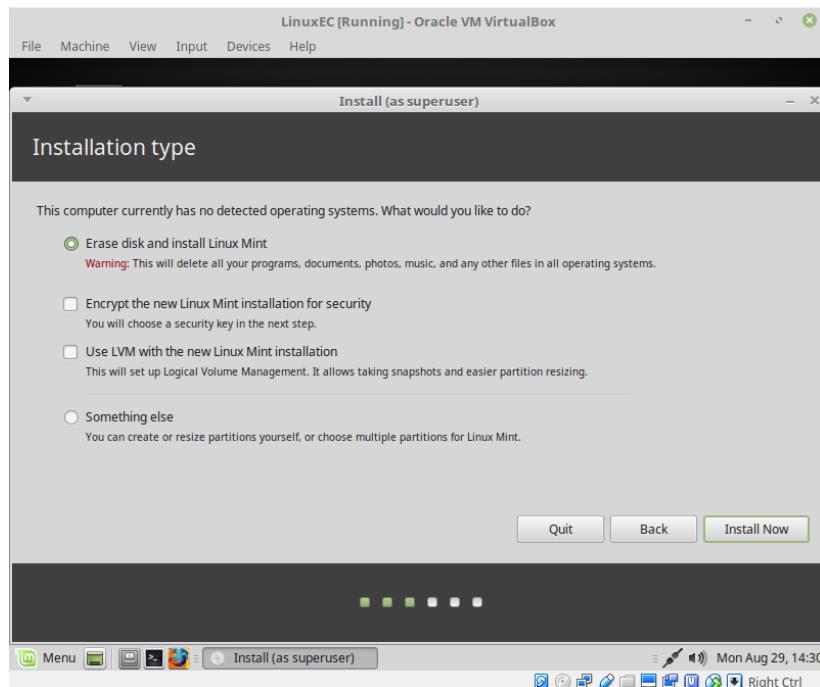


Continueu amb la instal·lació. Assegureu-vos d'estar connectats a Internet, i si esteu fent la instal·lació en un portàtil comproveu que esteu connectats a la corrent. Premeu el botó *Continue*.

En la pantalla següent seleccioneu l'opció d'instal·lar software de tercers. Premeu el botó *Continue*.

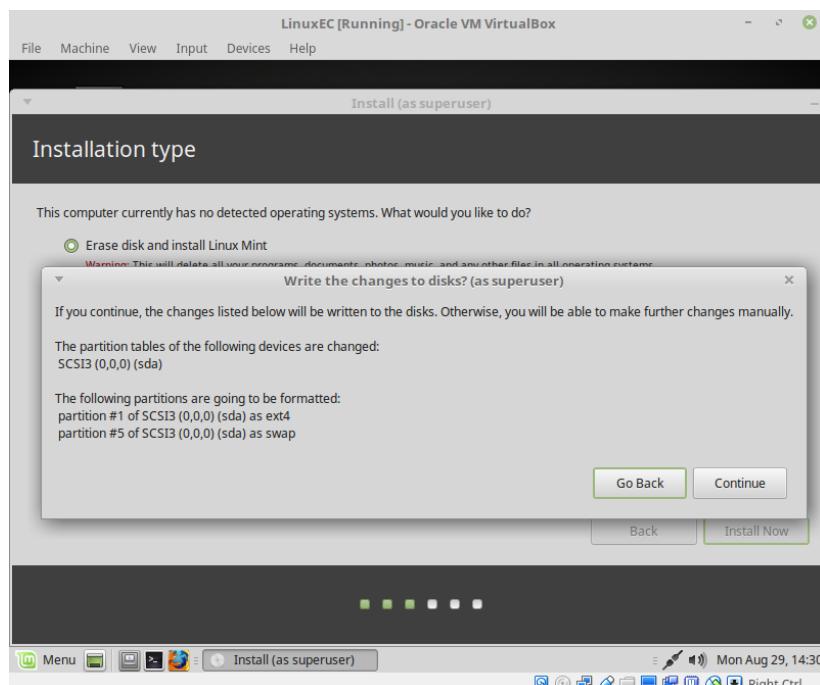


En la pantalla següent escolliu l'opció *Erase disk and install Linux Mint*. Aquesta operació només esborra el disc de la màquina virtual, no afecta al vostre ordinador.

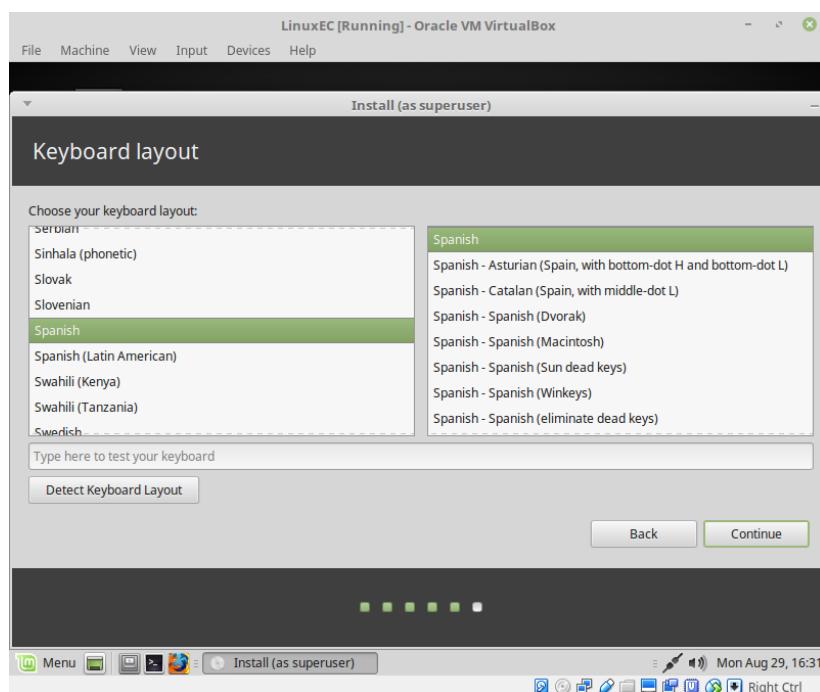
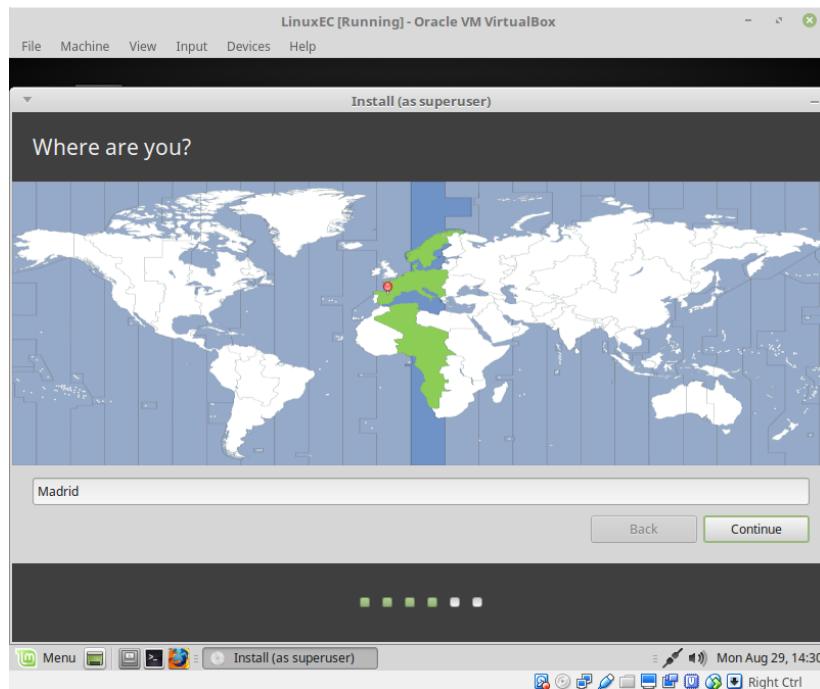


Premeu el botó *Install Now*.

Apareix una finestra de confirmació, on demana si volem escriure els canvis al disc virtual, premeu el botó *Continue*.

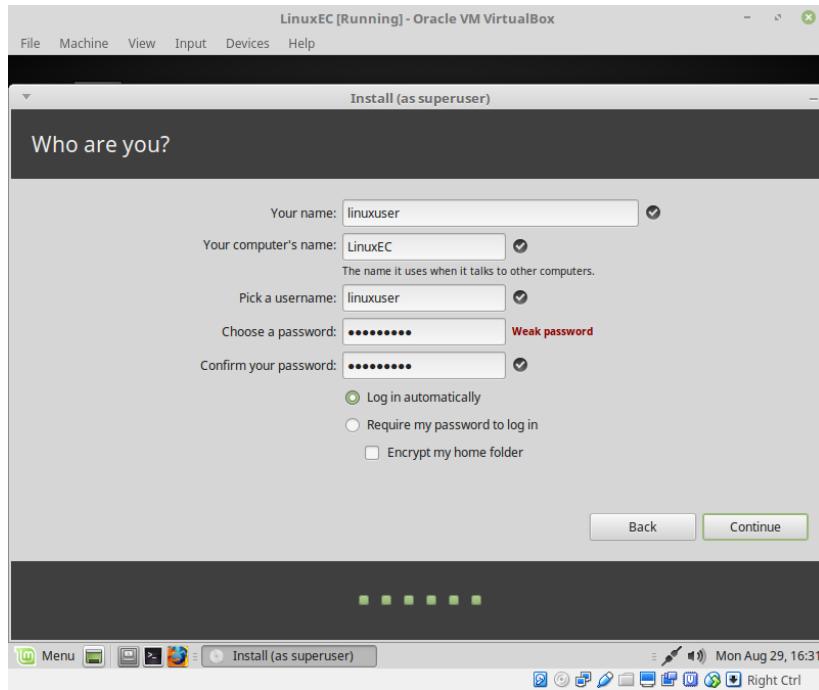


En les pantalles següents escolliu la vostra ubicació (fus horari), i el tipus de teclat.

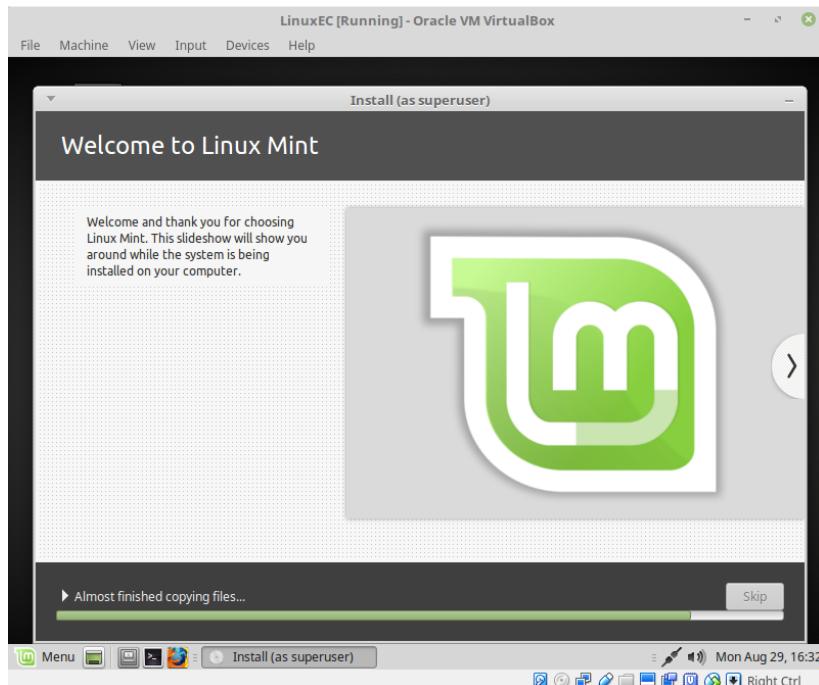


A continuació introduïu el vostre nom, un nom per a l'ordinador, escolliu un nom d'usuari i una contrasenya. És necessari omplir aquests camps. Cal recordar el nom d'usuari i la contrasenya introduïts, ja que seran necessaris quan haguem de realitzar tasques administratives i de configuració de la màquina virtual.

Podeu escollir l'opció *Log in automatically* (*entra de manera automàtica*), d'aquesta forma no us caldrà escriure el nom d'usuari i la contrasenya cada cop que inicieu la màquina virtual.



Podeu ignorar l'avertència sobre la contrasenya en el cas que hagueu posat una contrasenya molt senzilla. Premeu *Continue* i començarà la instal·lació. Espereu fins a què finalitzi.

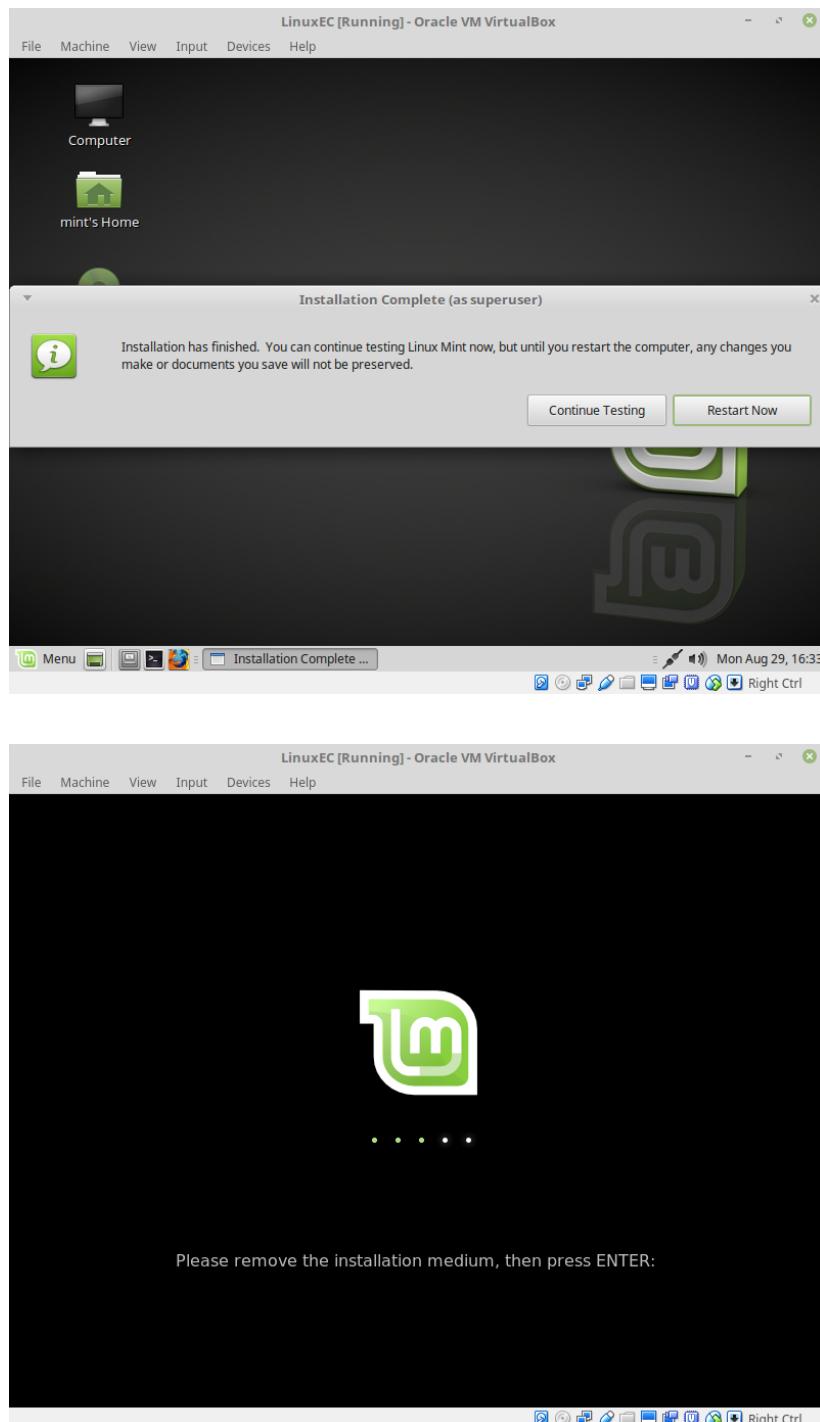


Finalment us demanarà si voleu reiniciar el sistema, escolliu *Restart Now*.

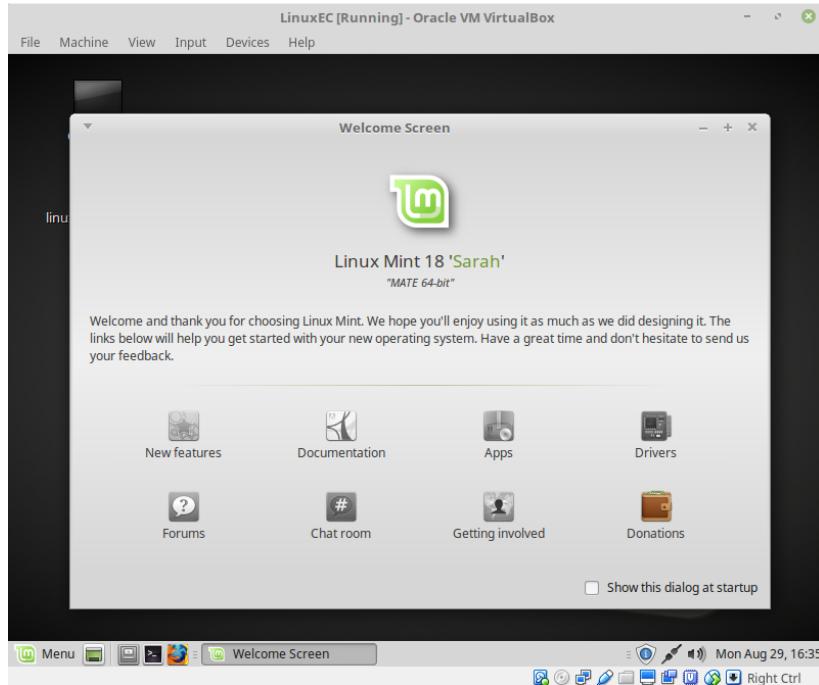
El sistema demana prémer la tecla ENTER, premeu la tecla, i el sistema s'hauria de reiniciar automàticament.

Atenció:

Si no veieu el missatge que demana prémer la tecla ENTER i el sistema es queda aturat amb la pantalla amb un fons negre o el fons de l'escriptori, premeu igualment la tecla ENTER estant sobre la pantalla de VirtualBox, el sistema s'hauria de reiniciar automàticament. Si no us funciona, tanqueu directament la finestra i escolliu l'opció "Power off the machine" (atura la màquina), a continuació inicieu manualment la màquina virtual.



Després de reiniciar el sistema, hauria d'entrar automàticament amb l'usuari creat i mostrar la pantalla inicial. Si no voleu tornar a veure aquesta pantalla desmarqueu l'opció de la cantonada inferior dreta: *Show this dialog at startup*.

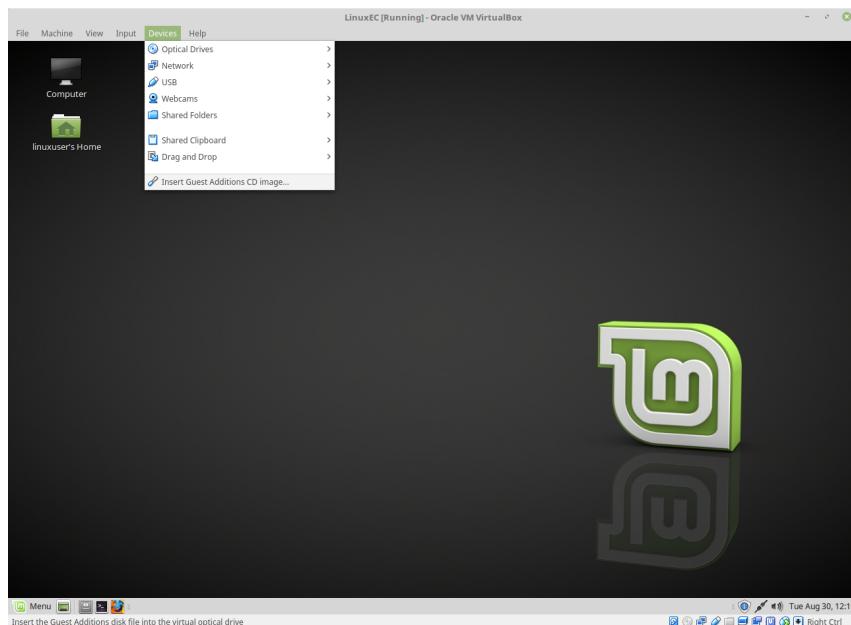


1.5. Instal·lar les Guest Additions de VirtualBox

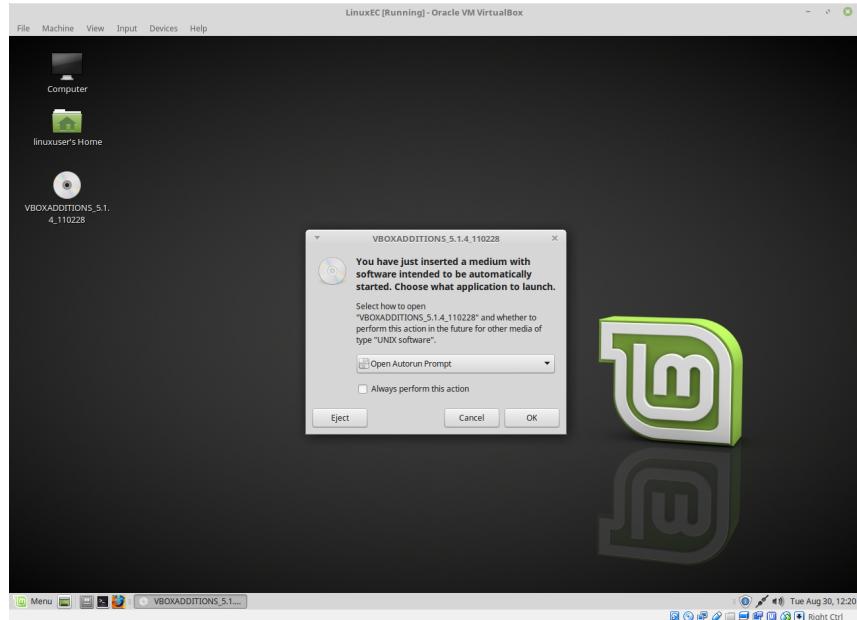
Cal instal·lar les Guest Additions per, entre altre coses per permetre redimensionar la finestra de la màquina virtual o compartir carpetes entre la màquina amfitriona i la màquina virtual.

Encara que les distribucions de Linux Mint ja proporcionen una versió de les Guest Additions, us recomanem instal·lar la versió de les Guest Additions corresponent a la versió de VirtualBox que esteu executant.

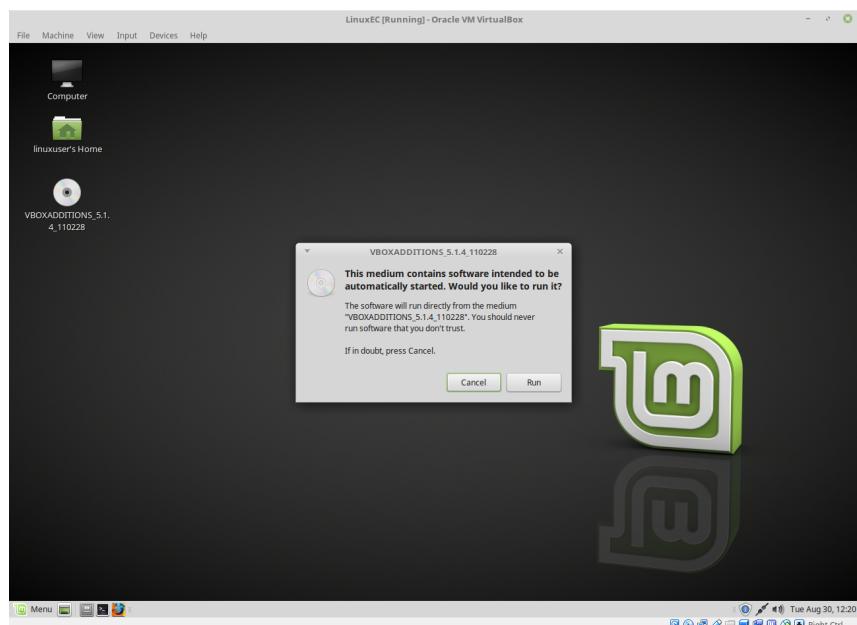
Per fer aquesta instal·lació seleccioneu l'opció *Insert Guest Additions CD image* del menú *Devices*.



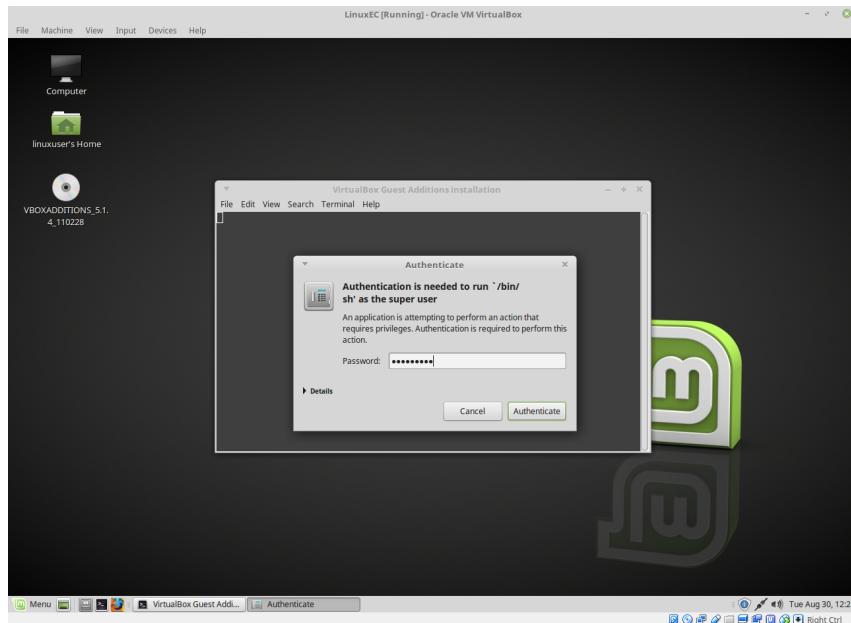
Us apareixerà una finestra d'execució automàtica. Premeu el botó *OK*.



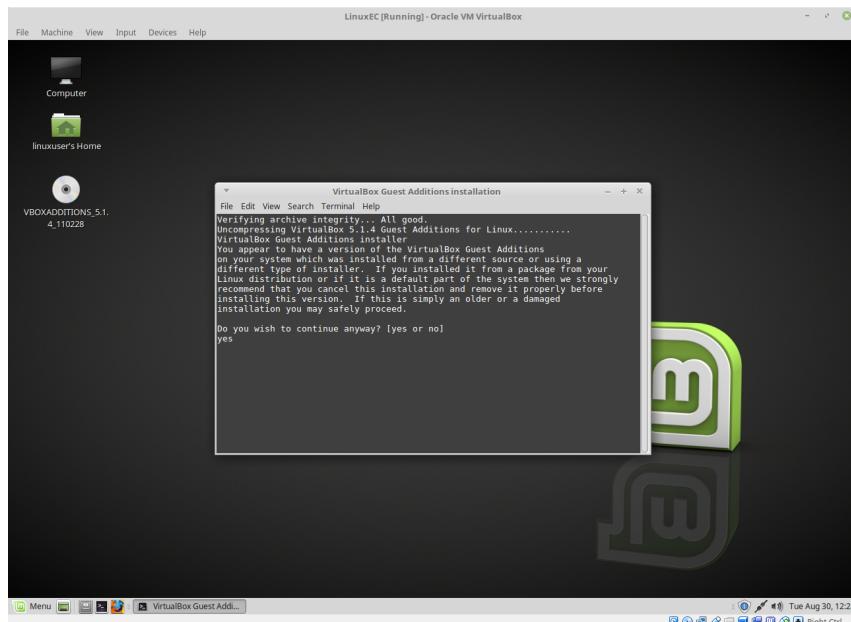
En la finestra següent premeu el botó *Run*.



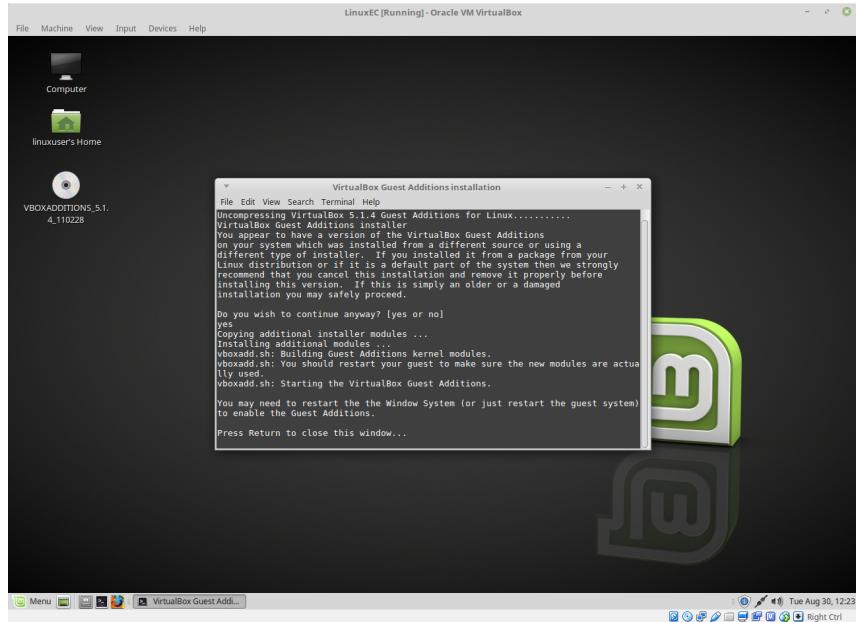
A continuació us demanarà la contrasenya de l'usuari de linux creat durant la instal·lació de Linux Mint, escriviu-la i premeu el botó *Authenticate*.



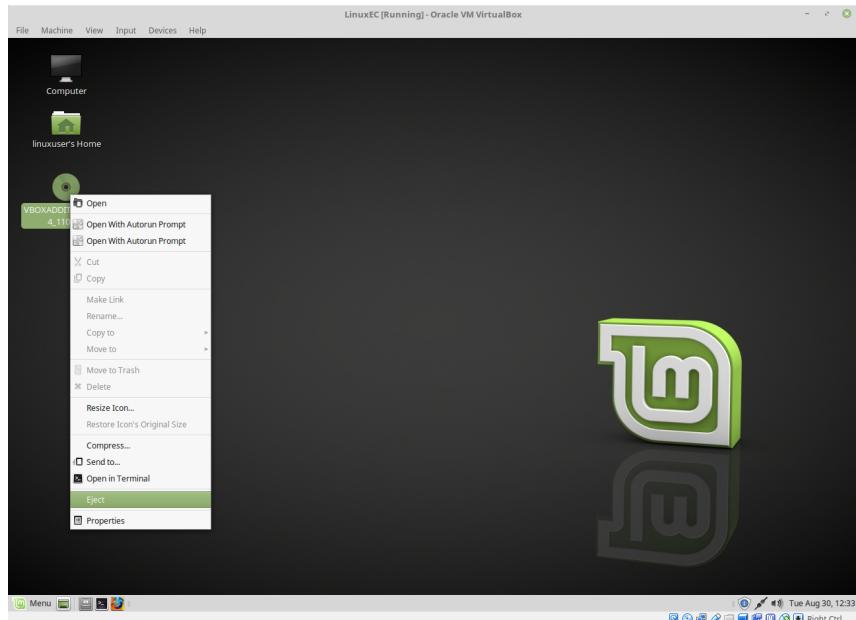
Apareixerà un missatge demanant confirmació, escriviu 'yes' i premeu ENTER.



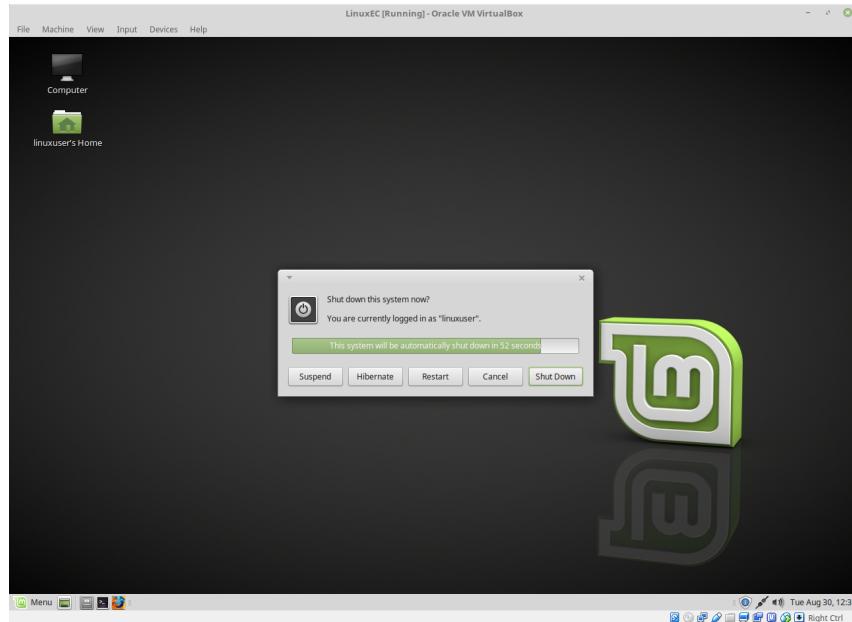
Quan finalitzi la instal·lació premeu Return (Enter) per tancar la finestra.



Podeu eliminar el disc virtual de les Guest Additions fent clic amb el botó dret del ratolí sobre la icona del CD de l'escriptori i escollint l'opció *Eject*.



Finalment reiniciu el sistema per activar els canvis, a través del *Menu* de la part inferior esquerra de Linux Mint amb l'opció *Quit → Restart*.

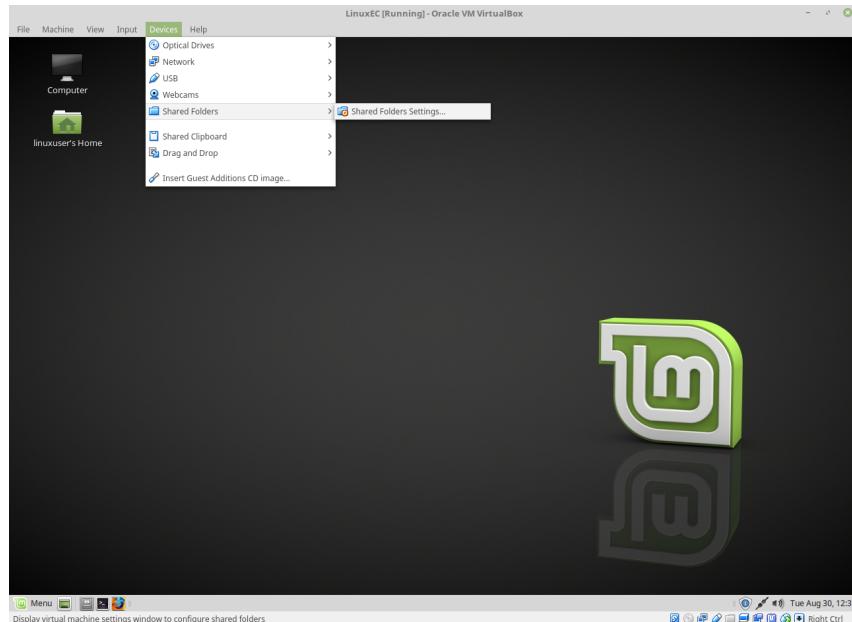


1.6. Carpetes compartides

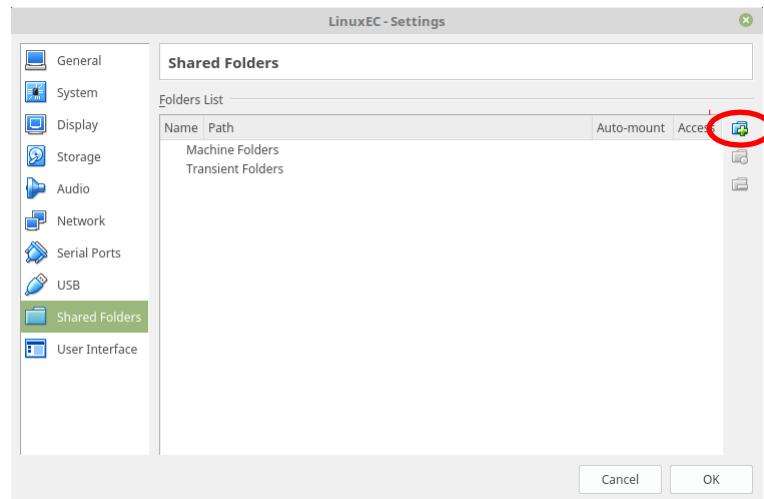
Podeu compartir una carpeta entre la màquina virtual i la màquina amfitriona, de forma que podreu accedir als fitxers d'aquesta carpeta des de dins i des de fora de la màquina virtual.

Per crear una carpeta compartida seguiu el procés següent.

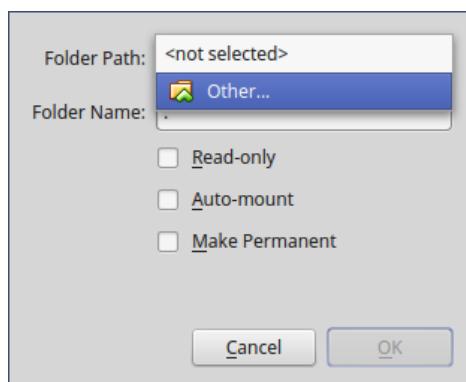
En la finestra de la màquina virtual obriu el menú *Devices* de VirtualBox, escolliu l'opció *Shared Folders → Shared Folders Settings*



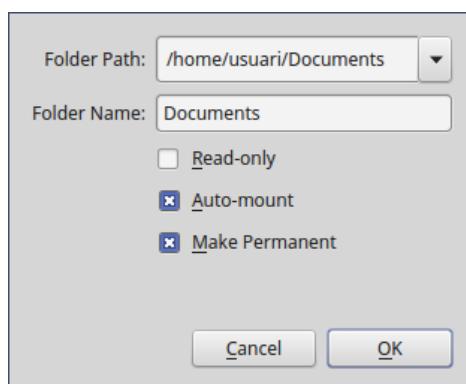
En la finestra que apareixerà premeu la icona de la carpeta amb un signe més de la part dreta per crear una nova carpeta compartida.



En la nova finestra que apareixerà escolliu l'opció *Other* en el desplegable *Folder Path*, navegueu per sistema de fitxers fins a la carpeta que voleu compartir.



En les opcions que apareixen seleccioneu: *Auto-mount* i *Make Permanent*. Accepteu els canvis.



Cal afegir l'usuari de la màquina virtual al grup de Linux vboxsf, per fer-ho seguiu els passos següents:

1. Obriu un terminal (Menu → *Terminal*)
2. Executeu l'ordre següent, substituïu “linuxuser” pel nom d'usuari que hagreu utilitzat durant la instal·lació. Us demanarà la contrasenya del vostre usuari per a realitzar tasques administratives. Recordeu que la contrasenya no es mostra per pantalla, premeu ENTER després d'introduir-la.

```
$ sudo usermod -a -G vboxsf linuxuser
```

Atenció

Cal respectar les majúscules i minúscules a l'escriure l'ordre. Us demanarà que introduïu la contrasenya de l'usuari.

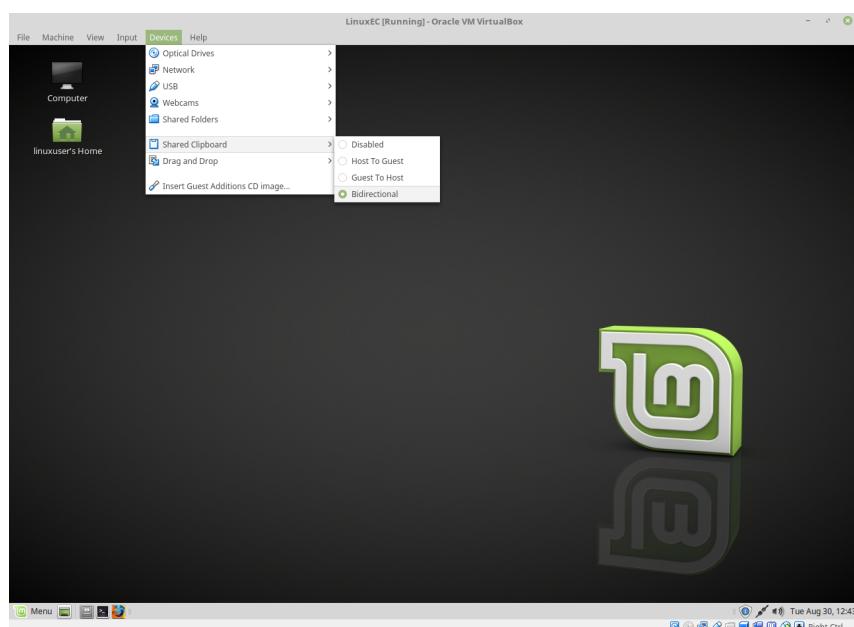
Aneu amb compte amb copiar les ordres des d'un document pdf, com aquest, directament en el terminal de Linux, es probable que els caràcters especials com el guions no es copiïn bé, i l'ordre no us funcioni

3. Reinicieu la màquina virtual. La nova carpeta compartida hauria d'aparèixer muntada en el directori */media/sf_CarpetaCompartida*, en el cas de l'exemple anterior seria: */media/sf_Documents*. En el explorador de fitxers de Linux accediu a través de Sistema de fitxers, carpeta *media*.

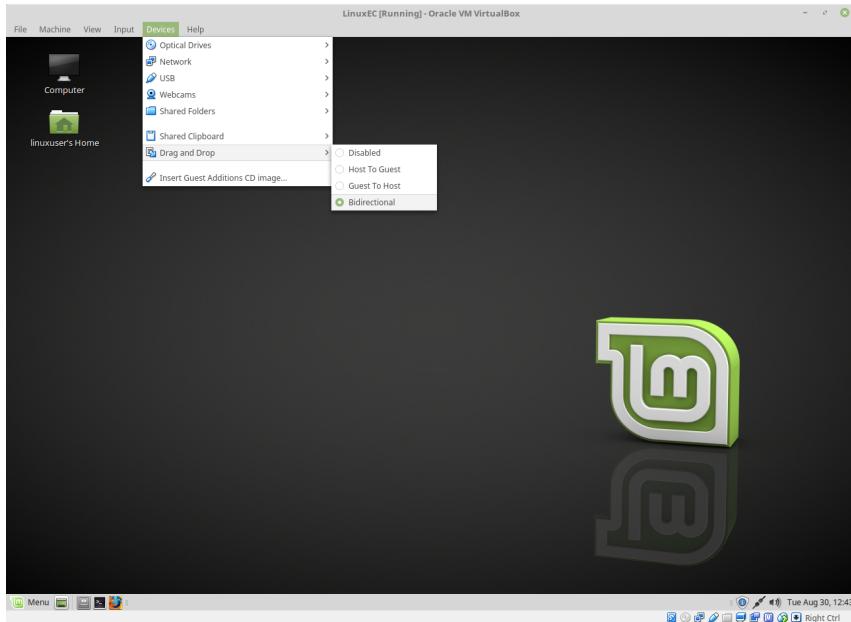
1.7. Porta-retalls compartit i arrossegar fitxers

Podeu compartir el porta-retalls entre la màquina virtual (client) i la màquina amfitriona, de forma que podem copiar i enganxar contingut entre les dues màquines.

Per activar la compartició del porta-retalls, accediu al menú *Devices* → *Shared Clipboard* de VirtualBox, i escolliu l'opció *Bidirectional*.



També podem arrossegar fitxers entre la màquina client i la màquina amfitriona, per activar l'arrossegament de fitxers accediu al menú *Devices → Drag and Drop* de VirtualBox, i escolliu l'opció *Bidirectional*.



1.8. Accedir a un dispositiu USB

Si heu instal·lat el paquet de VirtualBox, Oracle Extension Pack, podeu accedir als dispositius USB des de dins de la màquina virtual.

Per fer-ho accediu al menú de VirtualBox *Devices → USB*, i seleccioneu el dispositiu desitjat, es muntarà automàticament i apareixerà a l'escriptori de Linux.

1.9. Instal·lació de les eines

A continuació explicarem una manera d'instal·lar les eines en un entorn Linux Mint (també és vàlid per Ubuntu).

Per facilitar la tasca us proporcionem un script que realitza la tasca d'instal·lació per vosaltres i és la manera que us recomanem que feu servir.

Script install-EC.sh

Trobareu l'script juntament amb aquest document al tauler de l'assignatura. Podeu descarregar-lo dins la màquina virtual accedint amb el navegador Firefox al campus virtual. Si l'heu descarregat en la vostra màquina amfitriona el podeu copiar dins de VirtualBox utilitzant l'arrossegament de fitxers entre la màquina amfitriona i la màquina virtual o a través d'una carpeta compartida.

Descarregueu l'script dins la màquina virtual i a continuació executeu-lo des d'un terminal (*Menú → Terminal*).

Per fer-ho primer canviem al directori on es troba l'script, i a continuació l'executem. Suposant que l'script es troba a l'escriptori (/home/usuari/Desktop) de l'usuari executem les ordres següents:

```
$ cd /home/usuari/Desktop
$ sh install-EC.sh
```

Us demanarà la contrasenya del vostre usuari per a realitzar tasques administratives. Recordeu que la contrasenya no es mostra per pantalla, premeu ENTER després d'introduir-la.

Es mostraran en el terminal les ordres que realitza espereu fins a què finalitzi i mostri de nou el símbol del sistema. Tot el procés pot trigar alguns minuts ja que s'han de descarregar alguns paquets a través d'Internet, assegureu-vos de tenir connexió a Internet abans i durant l'execució de l'script.

Si durant la instal·lació us donés algun error, simplement torneu-lo a executar.

Un cop instal·lats tots els paquets podreu treballar-hi executant-los des d'un terminal.

També podreu accedir a les eines que disposen d'una interfície gràfica des del menú d'aplicacions.

L'editor hauria d'aparèixer en el menú

Menú → Applications → Programming → Geany

L'entorn de depuració hauria d'aparèixer en el menú

Menú → Applications → Programming → KDbg

De tota manera és més pràctic executar el *kdbg* des d'un terminal, ja que ens permet indicar el nom de l'executable que volem depurar i localitza de forma automàtica el fitxer amb el codi font.

1.9.1. Instal·lació manual de les eines

No cal realitzar aquests passos si heu utilitzat l'script, install-EC.sh

Per a instal·lar qualsevol programari cal tenir privilegis de superusuari, per aquest motiu cal utilitzar l'ordre *sudo* i introduir la contrasenya de l'usuari quan calgui.

Instal·lació manual del programari

Per a instal·lar el programari necessari ho farem des d'un terminal.

Obriu un terminal (*Menú → Terminal*) i executeu l'ordre següent:

```
$ sudo apt install -y geany geany-plugins yasm xterm kdbg build-essential
```

Els paquets gcc i gdb ja es troben instal·lats en Linux Mint, si utilitzeu una altra distribució pot ser necessari afegir els dos paquets a l'ordre anterior.

Configurar kdbg manualment

Cal configurar l'entorn de depuració gdb/kdbg per a que es mostri el codi assemblador generat per als programes escrit en C en notació Intel.

Per a fer-ho cal seguir els passos següents:

1. Obriu un terminal (*Menú → Terminal*) i executeu l'ordre següent, per iniciar l'editor de text *geany* o inicieu-lo des del menú d'aplicacions (*Menú → Applications → Programming → Geany*):

```
$ geany /home/usuari/.gdbinit
```

Substituïu "usuari" pel vostre nom d'usuari, fixeu-vos que el nom del fitxer .gdbinit comença amb un punt.

2. Escriviu una línia de text amb el contingut següent:

```
set disassembly-flavor intel
```

3. Graveu el fitxer i tanqueu l'editor *geany*.

4. Si teniu problemes a l'iniciar *kdbg*, amb els fitxers d'ícones de l'aplicació, podeu eliminar el fitxer que conté una icona de l'aplicació, amb l'ordre següent:

```
$ sudo rm /usr/share/kde4/apps/kdbg/icons/hicolor/22x22/actions/pulse.mng -f
```

Escriviu tota l'ordre en una sola línia.

5. Inicieu *kdbg*, ho podeu fer des del terminal:

```
$ kdbg
```

Accediu al menú *Settings → Global options*

En la línia: *How to invoke GDB*, modifiqueu el valor que apareix eliminant el paràmetre *-nx*, deixeu només:

```
gdb --fullname
```

6. El proper cop que inicieu *kdbg* el codi assemblador es mostrerà en notació Intel.

Crear un fitxer de ressaltat assemblador per a geany manualment

Obriu un terminal (*Menú → Terminal*) y executeu l'ordre següent:

```
$ mkdir $HOME/.config/geany/filedefs -p
```

Aquesta ordre crearà un directori anomenat *filedefs* dins del directori d'usuari.

Trobareu un fitxer amb el nom *filetypes.asm* en el tauler general de l'assignatura, juntament amb aquest mateix document, descarregueu-lo dins de la màquina virtual per a poder-lo copiar, per exemple a Desktop (escriptori).

Executeu l'ordre següent:

```
$ cp $HOME/Desktop/filetypes.asm $HOME/.config/geany/filedefs
```

L'ordre copia el fitxer *filetypes.asm* del *Desktop* de l'usuari al directori *filedefs* dins del directori de l'usuari, substituïu *Desktop* pel directori on hagueu descarregat el fitxer *filetypes.asm*

2. Utilització de les eines

2.1 Editor de text: geany

Per a escriure codi en llenguatge C o Assemblador es pot utilitzar qualsevol editor de text. Alguns editors disposen de ressaltat de la sintaxi, característica que mostra els elements del llenguatge utilitzant diferents colors per a cada un d'ells i altres funcionalitats que ajuden a programar. Existeixen diferents editors amb característiques semblants. En aquest document es descriu la utilització de l'editor **geany**, però se'n poden utilitzar d'altres.

Per a obrir l'editor s'hi pot accedir pel menú, *Applications → Programming* o en mode línia des d'un terminal executant l'ordre *geany*, afegint opcionalment com a paràmetre el nom del fitxer amb el codi font (fitxer *.asm* o *.c*) que es vol editar, si no existeix el crea.

Terminal Linux

En Linux hem d'executar molts dels programes i eines des d'un terminal, una línia d'ordres.

En Linux Mint podeu accedir al terminal (o consola) a través de l'accés directe del menú sota el grup System i a Favorites: *Terminal*. I també des del menú *All Applications → System Tools → Terminal*.

Si feu clic amb el botó dret sobre Terminal, us apareix un menú d'opcions que us permet afegir un accés directe a l'escriptori o al tauler inferior

```
$ geany program1.asm
```

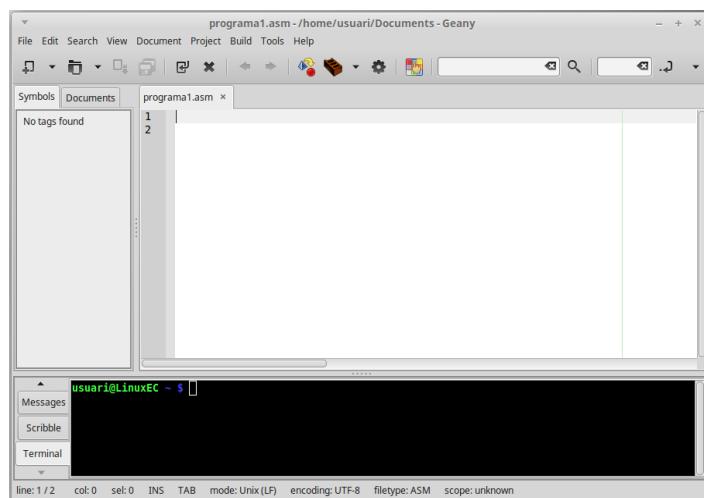
En executar l'editor s'obre una finestra amb la interície per a treballar amb aquesta eina.

Geany integra l'accés a un terminal dins de la mateixa aplicació; aquest terminal permet executar ordres des de l'editor mateix. Per a obrir el terminal només es necessari prémer el botó *Terminal* que es troba a la part inferior esquerra de la finestra de l'editor.

Si no veieu el botó feu la finestra de Geany una mica més gran o moveu-vos amb les fletxes amunt i avall de la part inferior esquerra.

Directori del terminal de Geany

Podeu fer que el directori actual del terminal sigui el mateix que el directori del fitxer actual que estigueu editant, per activar aquesta opció, feu el següent: accediu al menú de Geany: *Edit → Preferences*, en la pestanya *Terminal* marqueu l'opció *Follow path of the current file*, premeu OK per acceptar els canvis.



2.2. Assemblador: yasm

Un assemblador és un programa que a partir de codi font (codi assemblador) generar codi objecte. Hi ha diferents assembladors amb sintaxi Intel per sistemes Linux. Com cada assemblador utilitza una sintaxi diferent, s'ha escollit un en concret per treballar els exercicis pràctics. En aquest document es descriu la utilització de *yasm*.

Yasm es una nova versió completa de l'assemblador NASM (Netwide Assembler).

Actualment suporta el joc d'instruccions de les arquitectures x86 i x86-64, accepta les sintaxis NASM i GAS, genera codi objecte en els formats següents: ELF32, ELF64, Mach-O, RDOFF2, COFF, Win32, i Win64, i genera informació de depuració en els formats STABS, DWARF 2, i CodeView 8.

Documentació yasm

Podeu trobar informació detallada de yasm al seu web oficial: <http://yasm.tortall.net/>

Amb l'ordre de línia de Linux `man yasm` podeu consultar ràpidament tots els paràmetres i algunes qüestions bàsiques sobre la sintaxi que es poden utilitzar amb yasm.

Algunes de les característiques de Yasm que fan interessant el seu us són:

- Multiplataforma: hi ha versions de Yasm per Linux i Windows
- Sintaxi: utilitzà la sintaxi d'Intel, aquesta sintaxi és la més utilitzada i en podeu trobar molta documentació.

És un programa que funciona en mode línia i que s'ha d'executar des d'un terminal. Per a executar-lo, s'ha d'escriure l'ordre `yasm`, afegint els paràmetres necessaris.

Els paràmetres mínims que necessitarem per a poder generar codi objecte per un entorn Linux de 64 bits són els següents:

<code>nom_fitxer_font</code>	Nom del fitxer amb el codi font; recomanem utilitzar l'estensió <code>.asm</code> en aquests fitxers per a facilitar-ne la identificació.
<code>-f elf64</code>	Paràmetre per a indicar el format del codi objecte que genera; en sistemes Linux de 64 bits el format ha de ser “elf64”.

Opcionalment es poden especificar altres paràmetres com per exemple:

<code>-g dwarf2</code>	Paràmetre per a què generi informació per a la posterior depuració del programa; DWARF2 és un format d'informació de depuració que està fortament associat al format de codi objecte ELF.
<code>-o nom_fitxer_objecte</code>	Paràmetre per a especificar el nom del fitxer de codi objecte que es genera. Si no es posa aquest paràmetre el fitxer de codi objecte generat s'anomena igual que el fitxer amb el codi font canviant l'estensió <code>.asm</code> per l'estensió <code>.o</code> o afegint-la si no en té. Es recomana utilitzar aquesta extensió per a facilitar-ne la identificació.

```
$ yasm -f elf64 -g dwarf2 programal.asm
```

Si en el procés d'assemblatge no es produeixen errors, Yasm genera un fitxer amb el codi objecte, si es produeixen errors, aquests es mostren en pantalla amb el número de línia on s'ha detectat i una breu descripció, i no es genera el fitxer amb el codi objecte. En el procés de desenvolupament s'explica amb un exemple com interpretar els errors.

2.3. Enllaçador: gec (ld)

L'enllaçador és un programa que a partir del codi objecte generat per l'assemblador genera codi executable.

L'enllaçador del sistema és `ld`, però nosaltres utilitzarem el programa `gcc` que internament és capaç de cridar a `ld`, no farem crides explícites a `ld`, `gcc` és un programa que funciona en mode línia i s'ha d'executar des d'un terminal; per a executar-lo s'ha d'escriure l'ordre `gcc`, i afegir els paràmetres necessaris.

Els paràmetres mínims que necessitarem per a poder generar codi executable a partir d'un codi objecte són els següents:

<code>nom_fitxer_objecte</code>	Nom del fitxer amb el codi objecte generat amb l'assemblador; si no s'ha especificat de manera diferent és el fitxer amb l'estensió <code>.o</code>
---------------------------------	---

Opcionalment es poden especificar altres paràmetres com per exemple:

<code>-o nom_fitxer_executable</code>	Paràmetre per a especificar el nom del fitxer de codi executable que es genera. Si no es posa aquest paràmetre, el fitxer de codi executable generat s'anomena <code>a.out</code>
---------------------------------------	---

```
$ gcc -o programal programal.o
```

Si en el procés d'enllaçat no es produueixen errors, es genera un fitxer amb el codi executable, si es produueixen errors aquests es mostren en pantalla amb una breu descripció, i no es genera el fitxer amb el codi executable, tot i que no és gens habitual que hi hagi errors en aquest procés.

2.4. Compilador de C: gcc

El compilador de C (*gcc*), a més de compilar codi C per a generar el codi objecte, es capaç de cridar al programa enllaçador del sistema (*ld*) per a generar el fitxer executable final a partir dels diferents codis objecte generats amb l'assemblador i el codi objecte obtingut a partir del codi C.

gcc és un programa que funciona en mode línia i s'ha d'executar des d'un terminal; per a executar-lo s'ha d'escriure l'ordre *gcc*, i afegir els paràmetres necessaris.

Els paràmetres mínims que necessitarem per a l'execució correcta del compilador de C són els següents:

nom_fitxers_c	Llista de fitxers de codi C; els noms de fitxer amb el codi font C, han de tenir l'extensió .c
---------------	--

Opcionalment es poden especificar altres paràmetres com per exemple:

fitxers_o	Llista de fitxers de codi objecte; si no s'ha especificat de manera diferent seran els fitxers amb l'extensió .o
-o fitxer_sortida	Paràmetre per a especificar el nom del fitxer de codi executable que es genera.
-g	Paràmetre perquè generi informació per a la depuració posterior del programa.

```
$ gcc -g -o programa2 programal.o programa2.c
```

Si en el procés de compilació no es produueixen errors, es genera un fitxer amb el codi executable; si es produueixen errors, es mostren en pantalla amb el número de línia on s'ha detectat i una breu descripció, i no es genera el fitxer amb el codi executable. En el procés de desenvolupament s'explica amb un exemple com interpretar els errors.

Si no s'especifica cap fitxer de codi font C, només s'especifiquen fitxers amb codi objecte, es genera l'executable a partir d'aquests fitxers. D'aquesta manera podem generar codi executable a partir de fitxers objecte obtinguts amb l'assemblador, procés equivalent a l'execució de l'enllaçador (*ld*) expliat anteriorment.

```
$ gcc -g -o programal programal.o
```

2.5. Depurador : kdbg (gdb)

Un depurador és el programa necessari per a seguir l'execució d'un programa pas a pas i veure com evolucionen els valors de registres i variables; és una eina bàsica per a poder detectar i corregir errors en els nostres programes. Tenir un bon domini d'aquesta eina pot reduir dràsticament el temps de desenvolupament d'un programa.

El depurador *gdb* és un programa que funciona en mode línia i s'ha d'executar des d'un terminal. S'inicia mostrant un indicador d'ordres (*prompt*) per a introduir les ordres que ens permeten fer el seguiment de l'execució del codi pas a pas, però fer la depuració d'aquesta manera resulta força complex.

Per a facilitar aquest procés de depuració dels programes escrits en C i assemblador utilitzarem una interfície gràfica per al *gdb*: *kdbg*.

Per a executar aquesta interfície es por fer des del menú del sistema, *Applications* → *Programming* → *KDbg*, però us recomanem executar-lo des d'un terminal executant l'ordre *kdbg*, afegint com a paràmetre el nom del fitxer executable que es vol depurar.

```
$ kdbg programal
```

En executar el depurador, s'obrirà una finestra amb la interfície per a treballar amb aquesta eina.

Si no s'indica el nom del programa que es vol depurar caldrà obrir el fitxer executable des de la intereficie del programa.

Amb *kdbg* per a obrir el fitxer executable s'ha d'utilitzar del botó *Executable* de la barra d'eines o l'opció de menú *File → Executable*, navegar pels fitxers fins a trobar el correcte, i seleccionar el fitxer executable (*programa1*) i mai el fitxer amb el codi font (*programa1.asm*).

Cal tenir present que el depurador treballa sobre el codi executable del programa i, si detectem un error en el codi mentre fem el seguiment pas a pas, no el podrem modificar des d'aquest entorn; haurem de tornar a l'editor del codi font, fer les modificacions necessàries, guardar els canvis, generar un fitxer executable nou, carregar novament el fitxer executable en el depurador i tornar a provar si el problema s'ha solucionat.

En el procés de desenvolupament s'explica amb un exemple com utilitzar aquesta eina.

2.6. Execució

El fitxer amb el codi executable generat pel *gcc* es pot executar des d'un terminal, simplement especificant el nom del fitxer, afegint al davant “*./*” per indicar que es troba en el directori actual.

```
$ ./programa1
```

Camins en Linux

En Linux els camins (path) per defecte en què el sistema operatiu cerca un programa per a executar-lo, no inclouen mai el directori actual; per aquest motiu per a executar un programa cal posar *./* davant el nom del programa.

3. Procés de desenvolupament en assemblador

Per a comprovar que l'entorn de programació funciona correctament se seguiran les diferents fases del procés de desenvolupament a partir d'un codi font donat en assemblador. Ara no us preocupeu d'analitzar el codi, més endavant ja treballarem les directives, les instruccions i la declaració de dades en profunditat.

3.1. Edició del codi font

Cal accedir a l'editor *geany* a partir de l'opció de menú corresponent (*Applications → Programming → Geany*) o escrivint l'ordre *geany* des d'un terminal.

```
$ geany hola.asm
```

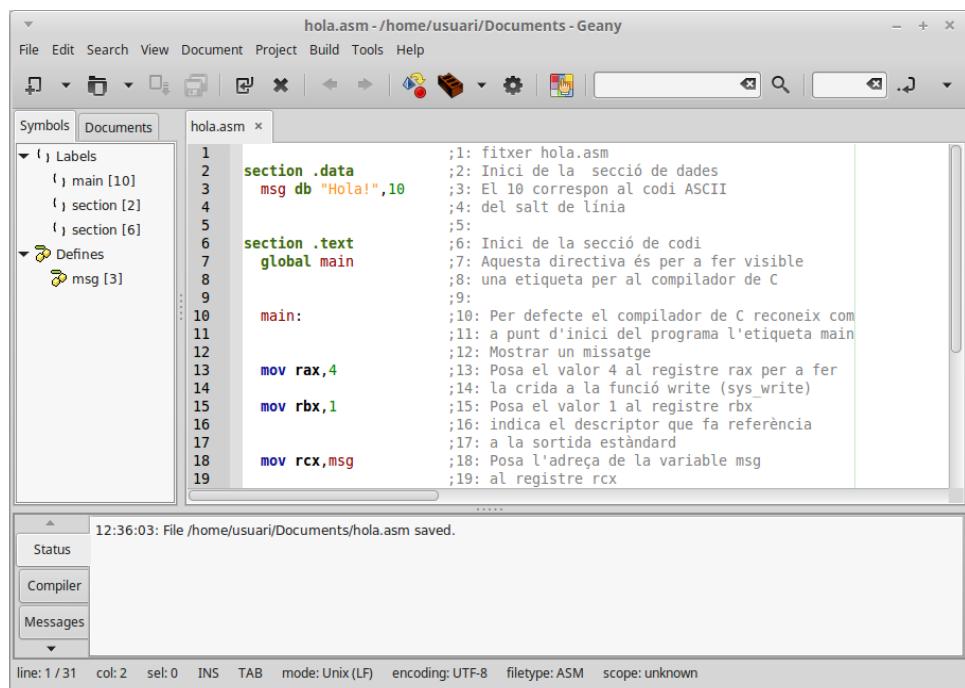
Codi font exemples

Podeu trobar tots els codis font dels exemples d'aquest document en el fitxer zip que conté aquest mateix document.

A la finestra de l'editor s'ha d'escriure el codi font, de 30 línies, següent:

```
;1: fitxer hola.asm
section .data
    msg db "Hola!",10
;2: Inici de la secció de dades
;3: El 10 correspon al codi ASCII
;4: del salt de línia
;5:
section .text
    global main
;6: Inici de la secció de codi
;7: Aquesta directiva és per a fer visible
;8: una etiqueta per al compilador de C
;9:
main:
;10: Per defecte el compilador de C reconeix com
;11: a punt d'inici del programa l'etiqueta main
;12: Mostrar un missatge
    mov rax,4
;13: Posa el valor 4 al registre rax per a fer
;14: la crida a la funció write (sys_write)
    mov rbx,1
;15: Posa el valor 1 al registre rbx
;16: indica el descriptor que fa referència
;17: a la sortida estàndard
    mov rcx,msg
;18: Posa l'adreça de la variable msg
;19: al registre rcx
    mov rdx,6
;20: Posa la longitud del missatge inclòs el 10
;21: del final al registre rdx
;22: crida al sistema operatiu
;23: retorna el control al terminal
;24: del sistema operatiu
    mov rax,1
;25: Posa el valor 1 al registre rax
;26: per a cridar a la funció exit (sys_exit)
    mov rbx,0
;27: Posa el valor 0 al registre rbx indica
;28: el codi de retorn (0 = sense errors)
    int 80h
;29: crida al sistema operatiu
;30:
```

Un cop escrit el codi, cal emmagatzemar el fitxer indicant un nom amb l'estensió .asm; *hola.asm* per exemple.



Geany reconeix alguns elements del llenguatge assemblador, instruccions, registres, definició de variables i etiquetes entre altres. Sota la pestanya *Symbols* us mostrarà les etiquetes i definicions que ha trobat en el codi font, permetent accedir directament a elles.

3.2. Assemblatge del codi font

Per a assemblar el codi cal executar des del terminal (podeu utilitzar el terminal de l'editor *geany*) l'ordre següent:

```
$ yasm -f elf64 -g dwarf2 hola.asm
```

També podeu utilitzar la icona Compila, o l'opció de menú *Build → Compile* de Geany.

Ordres de geany

L'editor Geany, permet definir accessos directes a les ordres d'assemblatge, muntatge, etc.

Podeu utilitzar l'opció de menú *Build → Set Build Commands* per personalitzar els accessos a les eines

Si no s'ha detectat cap error l'assemblador no mostrarà cap missatge, i s'haurà generat un fitxer amb el nom *hola.o* en el mateix directori on és el fitxer amb el codi font (*.asm*).

Si s'han detectat errors, es mostraran en pantalla juntament amb el número de línia on es troba l'error; cal modificar el codi per a corregir el errors indicats, guardar el codi modificat i tornar a assemblar el codi.

Fent els canvis següents al codi apareixeran 4 errors a l'assemblar-lo:

```

mov rcx,msg1 ;18: -> modifiquem el nom de l'etiqueta
mov rdx,      ;20: -> eliminem un operand
int 80h       ;22: -> modifiquem el codi d'operació

```

```
$ yasm -f elf64 -g dwarf2 hola.asm
hola.asm:18: error: undefined symbol `msg1' (first use)
hola.asm:18: error: (Each undefined symbol is reported only once.)
hola.asm:20: error: expected operand, got end of line
hola.asm:22: error: instruction expected after label
```

Amb aquesta informació hem de saber detectar quina és la causa de l'error o del *warning* i com corregir-lo, tot i que la informació que dóna el programa assemblador *yasm* no sigui gaire exhaustiva.

Si heu cridat a *yasm* des de l'opció de menú de Geany, els missatges apareixeran a la pestanya *Compiler* de la part inferior.

3.3. Enllaçament del codi objecte i generació de l'executabile

L'estapa següent consisteix a generar l'executabile utilitzant el compilador *gcc*; per a fer-ho cal executar des d'un terminal (es pot utilitzar el terminal integrat amb l'editor Geany) l'ordre següent:

```
$ gcc -o hola hola.o
```

Si el procés s'executa correctament es generarà un fitxer nou, en el mateix directori, amb el nom indicat pel paràmetre *-o*; si no es posa aquest paràmetre, el fitxer executable generat s'anomena *a.out*.

3.4. Execució del programa

Per executar el programa, cal escriure des del terminal (es pot utilitzar el terminal integrat amb l'editor Geany) el nom del executable i indicar el directori actual al davant:

```
$ ./hola
```

El programa s'hauria d'executar correctament i mostrar el missatge “Hola!”

```
$ ./hola
Hola!
$
```

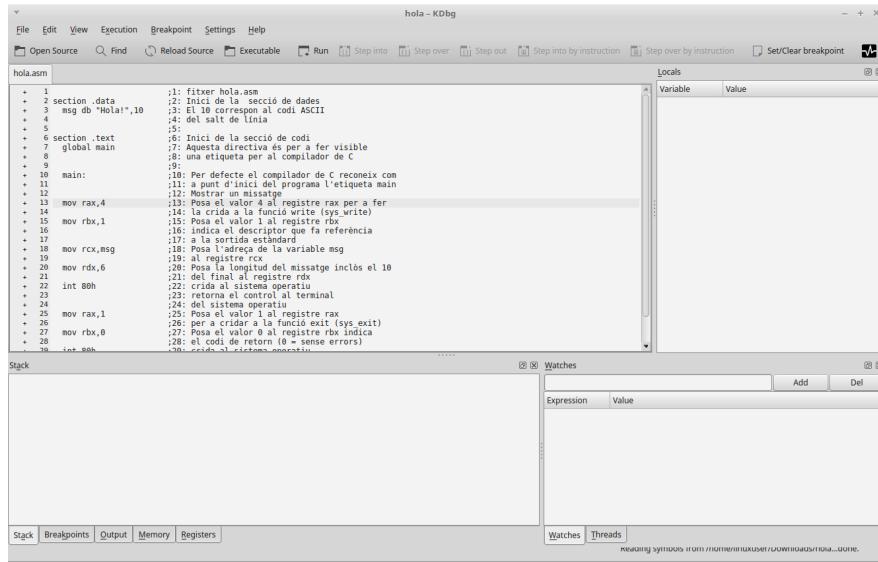
3.5. Depuració del programa amb KDbg (gdb)

A continuació s'explica la utilització de l'entorn de depuració KDbg.

Per a executar-lo cal executar des del terminal (es pot utilitzar el terminal integrat amb l'editor Geany) l'ordre següent:

```
$ kdbg hola
```

Si no s'indica el nom del programa que es vol depurar, caldrà obrir l'executabile des de la interfície del programa KDbg. S'ha d'utilitzar l'opció de menú *File → Executable*, navegar pels fitxers fins a trobar el correcte, i seleccionar el fitxer executable (*hola*) i no pas el fitxer amb el codi font (*hola.asm*).



Es descriuen a continuació les operacions més freqüents que es solen realitzar amb l'entorn KDbg.

1) Execució del programa

Per executar el programa tenim les opcions següents:

- Prém el botó *Run*
- Escollir l'opció de menú *Execution → Run*
- Prém la tecla F5.

El codi s'executa de cop fins el primer punt de ruptura (*breakpoint*); si no hem definit cap punt de ruptura s'executarà tot el codi, la sortida del programa per pantalla es farà a la finestra d'execució de l'entorn (*Program Output*).

Si no s'ha definit cap punt de ruptura, es fa l'execució completa del programa, i la sortida per pantalla serà la mateixa que si haguéssim executat el programa directament des del terminal.

Per a interrompre l'execució cal escollir l'opció de menú *Execution → Kill*; per reiniciar l'execució des del principi *Execution → Restart*.

Executeu el programa i comproveu que en la finestra d'execució es mostra el missatge: *Hola!*

2) Depurar el programa

Per a poder executar el codi pas a pas, primer cal definir un punt de ruptura en la instrucció a partir de la qual volem seguir l'execució pas a pas. Si volem depurar el codi des del principi, haurem de posar un punt de ruptura a la primera instrucció. Quan s'inicia l'execució del programa (*Run*), aquesta queda aturada en el primer punt de ruptura que troba, i podem seguir l'execució pas a pas a partir d'aquell punt.

Per a definir un punt de ruptura tenim les opcions següents:

- Prém amb el ratolí al principi de la instrucció (davant del signe +) on es vol definir el punt de ruptura, la línia quedarà marcada amb un punt de color vermell.
- Situar el cursor en la línia on es vol introduir el punt de ruptura i prém el botó *Set/Clear breakpoint* o prém la tecla F9

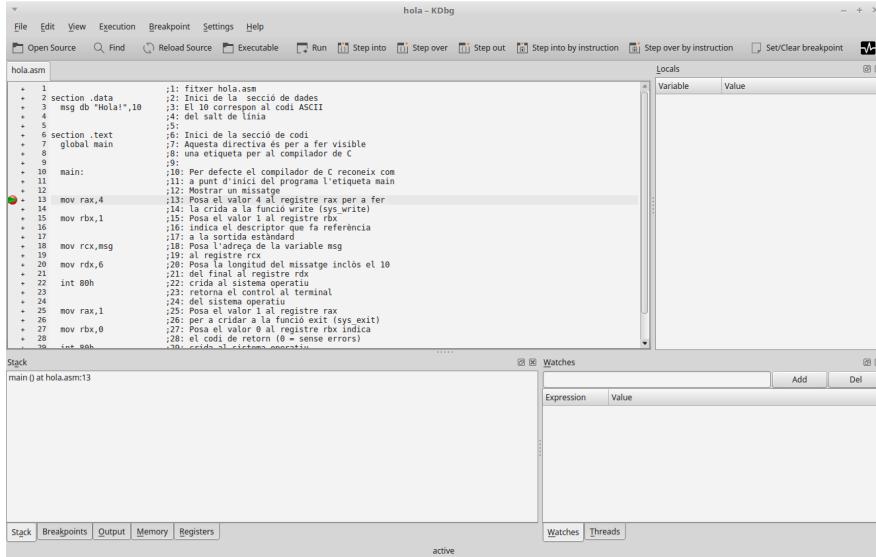
Per esborrar un punt de ruptura definit tenim les opcions següents:

- Tornar a prém amb el ratolí al principi de la instrucció on es troba el punt de ruptura, desapareixerà el punt de color vermell.
- Situar el cursor en la línia on és el punt de ruptura i prém el botó *Set/Clear breakpoint* o prém la tecla F9

Poseu-vos a la línia 13 (*mov rax, 4*), primera instrucció del programa, i definiu un punt de ruptura.

Executeu el programa (*Run*).

L'execució s'aturarà a la primera instrucció, instrucció on hem posat el punt de ruptura. Apareix un triangle verd al principi de la línia que indica que és la següent instrucció que serà executada (no s'ha executat encara). La finestra del depurador queda de la següent manera:



KDbg disposa d'una barra d'eines des d'on es poden realitzar les operacions més habituals de depuració, també es pot accedir a aquestes operacions a través de tecles de funció i a través del menú *Execution*.



Run: execució fins al punt de ruptura següent; F5 o *Execution → Run*.

Step into: execució pas a pas, entrant dins les funcions o subrutines; F8 o *Execution → Step into*.

Step over: execució pas a pas, sense entrar dins les funcions o subrutines; F10 o *Execution → Step over*.

Step out: sortir d'una funció o subrutina; F6 o *Execution → Step out*.

Step into by instruction: execució instrucció a instrucció, entrant dins les funcions o subrutines assemblador; Majúscules+F8 o *Execution → Step into by instruction*.

Step over by instruction: execució instrucció a instrucció, sense entrar dins les funcions o subrutines assemblador; Majúscules+F10 o *Execution → Step over by instruction*.

Si estem depurant codi assemblador, les operacions d'executar pas a pas (Step into) i executar instrucció a instrucció (Step into by instruction) faran el mateix, però si depurem codi C el comportament serà diferent.

Executeu el programa instrucció per instrucció fins que finalitzi.

3) Execució fins al cursor

Podem executar un conjunt d'instruccions de cop, des de la instrucció on està aturada l'execució (instrucció marcada amb un triangle verd) i fins la instrucció marcada per la posició del cursor prement la tecla F7.

Aquesta opció és molt pràctica perquè en permet anar fàcilment a un punt concret del codi per seguir des d'allà l'execució pas a pas.

4) Continuar l'execució

Si el programa es troba aturat en un punt de ruptura, podem continuar l'execució d'aquest fins al punt de ruptura següent tornant a premer l'opció *Run*.

5) Veure variables i registres

Podem visualitzar el valor de variables definides en el nostre programa, altres expressions o registres en la part *Watches* (a la part inferior dreta de la pantalla principal). Per visualitzar el contingut d'una expressió cal escriure'n el nom al quadre de text i premer el botó *Add*.

Podem utilitzar el botó *Del* per esborrar una expressió.

Podem escriure diferents tipus d'expressions per a la seva visualització:

- **nom_variable**: es mostrarà el contingut de la variable
- **&variable**: es mostrarà l'adreça de la variable i es podrà veure també el seu contingut
- **\$registre**: es mostrarà el contingut del registre de 64 bits indicat, per exemple \$rax, \$rbp o \$r9. També podem veure només una part d'un registre utilitzant el nom adequat.

Per veure una part d'un dels registres entre r8 i r15, cal utilitzar un sufix darrera del nom del registre de 64 bits:
 l (lletra 'l', byte menys significatiu)
 w (word: els dos bytes menys significatius)
 d (double word: els 4 bytes menys significatius)

Per als registres rax, rbx, rcx i rdx podem utilitzar:

al, bl, cl, dl per veure el byte menys significatiu
 ah, bh, ch, dh per veure el segon byte menys significatiu

Per als registres rsi, rdi, rbp podem utilitzar:

sil, dil, bpl per veure el byte menys significatiu

Per als registres rax, rbx, rcx, rdx, rsi, rdi, rbp podem utilitzar:

ax, bx, cx, dx, si, di, bp per veure els 2 bytes menys significatius.
 eax, ebx, ecx, edx, esi, edi, ebp per veure els 4 bytes menys significatius.

Cal posar els noms dels registres sempre en minúscules i sempre començant amb el signe \$.

Exemples:

\$al: es mostrarà el byte menys significatiu del registre RAX.
\$ah: es mostrarà el segon byte menys significatiu del registre RAX.
\$ax: es mostrarà el valor dels 2 bytes menys significatius del registre RAX.
\$eax: es mostrarà el valor dels 4 bytes, menys significatius, del registre RAX
\$rax: es mostrarà el valor del registre RAX sencer (8 bytes)
\$r8l: es mostrarà el byte menys significatiu del registre RAX
\$r8d: es mostrarà el valor dels 4 bytes, menys significatius, del registre R8.

Es pot forçar la mida de les dades a mostrar indicant un tipus de dada al davant de l'expressió, utilitzant char (1 byte), short (2 bytes), int (4 bytes) o long (8 bytes) i especificar quants valors de la mida indicada volem veure entre []

- **(char) variable**: es mostrarà el valor de la variable com un sol caràcter, 1 byte
- **(char[8]) variable**: es mostrarà els valors de 8 bytes a partir de l'adreça de la variable com a caràcters.
- **(int[2]) variable**: es mostrarà el valor de 2 enters de 32 bits.
- **(char[8]) \$rax**: es mostrarà el valor dels 8 bytes del registre RAX: char[0] correspondrà al registre AL, char[1] correspondrà al registre AH, etc.

Es pot forçar que els valors es mostrin en un determinat format afegint un prefix davant de l'expressió: /t binari, /d decimal amb signe, /u decimal sense signe, /x hexadecimal, /c caràcter:

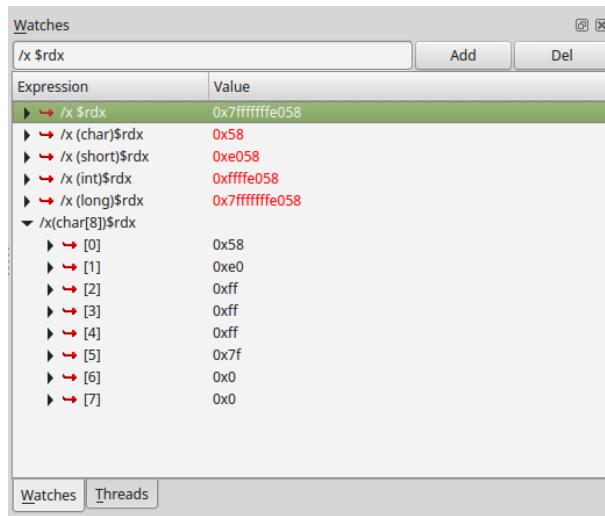
- **/c var**
- **/d var**
- **/u var**
- **/t \$rax**
- **/x \$rax**

Podem combinar els tipus de dades amb els prefix d'un cert format:

- **/u (char[8]) var**: es mostrarà la variable com a 8 bytes, cada byte es veurà com un nombre decimal sense signe.
- **/u (char[8]) \$rax**: igual que en el cas anterior, però mostrant els 8 bytes del registre RAX.
- **/x (char[8]) \$rax**: igual que en el cas anterior, però mostrant els 8 bytes del registre RAX en hexadecimal.

Si tenim una variable de tipus apuntador (una adreça de memòria) podem veure el seu contingut amb l'operador *

- **(* apuntador)**



Si tenim definit un vector definit en codi C, podem veure el seu contingut posant directament el nom del vector (ja que el nom ja representa un apuntador al primer element):

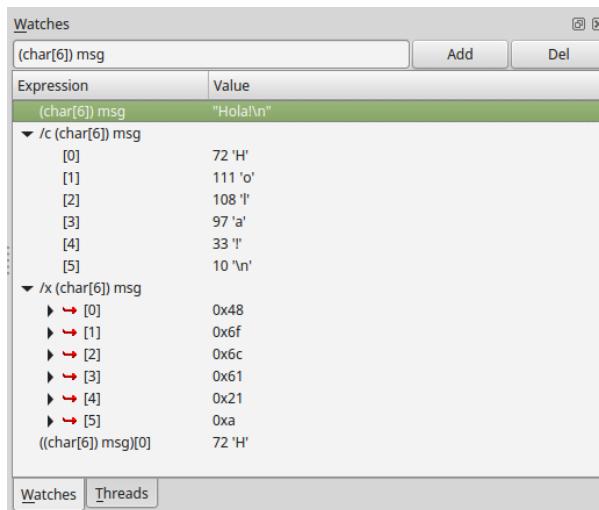
- **vector:** es mostraran els elements del vector.
- **/d vector:** es mostraran els elements del vector com a valors decimals.
- **/x vector:** es mostraran els elements del vector com a valors hexadecimals.
- **vector[0]:** es mostrarà el primer element del vector, índex 0.

Per veure un vector definit en assemblador cal indicar la seva mida al davant, en assemblador no es coneix la mida:

- **(char[6]) msg:** es mostraran 6 elements del vector com a caràcters de forma consecutiva.
- **/c (char[6]) msg:** es mostraran 6 elements del vector com a caràcters de forma separada.
- **/x (char[6]) msg:** igual que el cas anterior però com a valors hexadecimals.

Es pot especificar un índex per veure només un element

- **((char[6]) msg)[0]:** es mostrarà el primer element del vector, índex 0.



Quan es depura un programa escrit en C, a la part superior de la dreta de la pantalla es poden veure automàticament les variables locals definides i el valor que tenen.

6) Veure banc de registres

Podem veure el valors dels registres seleccionant la pestanya *Registers* a la part inferior de la pantalla.

Registers				
Register	Value	Decoded	value	
▼ GP and others				
rax	0x6	6		
rbx	0x1	1		
rcx	0x7fffff7b04a...	140737348913680		
rdx	0x7fffff7dd37...	140737351858048		
rsi	0x602010	6299664		
rdi	0x1	1		
rbp	0x7fffffffdf...	0x7fffffffdf60		
rsp	0x7fffffffdf...	0x7fffffffdf50		
r8	0x602000	6299648		
r9	0xd	13		
r10	0x7fffff7dd1b...	140737351850872		
r11	0x246	582		
r12	0x4004a0	4195488		
r13	0x7fffffff...	140737488347200		
r14	0x0	0		
r15	0x0	0		
rip	0x4005d0	0x4005d0	<main+50>	

Es poden veure els bits actius del registre d'estat EFLAGS prement amb el botó dret del ratolí a sobre del nom del registre *eflags* i seleccionant l'opció *GDB Default*.

7) Veure contingut de memòria

Prement la pestanya *Memory* de la part inferior de la pantalla es pot veure el contingut de la memòria a partir d'una adreça específica, es pot utilitzar el nom d'una variable o etiqueta posant al davant el símbol &, per exemple: &msg.

Si premem amb el botó dret del ratolí en aquesta finestra, ens apareix una llista d'opcions de visualització del contingut de la memòria, es recomana escollir l'opció *Bytes*, per veure el contingut byte a byte. També podem escollir el format de visualització: hexadecimal, decimal, caràcter, etc.

4. Procés de desenvolupament en C i assemblador

Ara veurem com utilitzar aquest entorn de programació per a desenvolupar programes a partir de codi font C i també com incorporar crides a codi font assemblador. Ara no us preocupeu d'analitzar el codi, més endavant ja treballarem les directives, les instruccions i la declaració de dades en profunditat.

En aquest procés, primer crearem el codi font assemblador i el deixarem a punt per poder-lo cridar després des del programa en C que farem a continuació.

4.1. Edició del codi font assemblador

Cal accedir a l'editor Geany a partir de l'opció de menú corresponent (*Applications → Programming → Geany*) o escrivint l'ordre *geany* des d'un terminal.

```
$ geany hola_asm.asm
```

A la finestra de l'editor s'ha d'escriure el codi font, de 25 línies, següent:

```
;1: fitxer hola_asm.asm
section .data ;2: Inici de la secció de dades
msg db "Hola!",10 ;3: El 10 correspon al codi ASCII
;4: del salt de línia
;5:
section .text ;6: Inici de la secció de codi
global printHola ;7: Aquesta directiva és per fer visible
;8: una etiqueta per al compilador de C
;9:
printHola: ;10: Nom que donem a la subrutina d'assemblador
;11: que cridarem des del programa en C
;12: Mostrar un missatge
mov rax,4 ;13: Posa el valor 4 al registre rax per a fer
;14: la crida a la funció write (sys_write)
mov rbx,1 ;15: Posa el valor 1 al registre rbx
;16: per indicar el descriptor que fa referència
;17: a la sortida estàndard
mov rcx,msg ;18: Posa l'adreça de la variable msg
;19: al registre rcx
mov rdx,6 ;20: Posa la longitud del missatge inclòs el 10
;21: del final al registre rdx
int 80h ;22: crida al sistema operatiu
;23:
ret ;24: retorn de la crida a subrutina.
;25:
```

Aquest codi és el molt semblant al codi *hola.asm* utilitzat en el punt anterior. Les modificacions que s'han fet són canviar l'etiqueta *main* per l'etiqueta *printHola*, afegir la instrucció *ret* per a tornar el control al programa que cridi aquesta subrutina, i eliminar les instruccions per retornar al sistema operatiu.

Un cop escrit el codi, deseu-lo indicant un nom amb l'estensió *.asm*, per exemple: *hola_asm.asm*

4.2. Assemblatge del codi font assemblador

Per assemblar el codi cal executar des del terminal (podeu utilitzar el terminal de l'editor Geany) l'ordre següent:

```
$ yasm -f elf64 -g dwarf2 hola_asm.asm
```

Si no s'ha detectat cap error, l'assemblador no mostrarà cap missatge, i s'haurà generat un fitxer amb el nom *hola_asm.o* en el mateix directori del fitxer amb el codi font (*.asm*).

Si s'han detectat errors, es mostraran en pantalla juntament amb el número de línia on es troba l'error; cal modificar el codi per a corregir els errors indicats, guardar el codi modificat i tornar a assemblar el codi.

4.3. Edició del codi font C

Cal accedir a l'editor Geany a partir de l'opció de menú corresponent (*Aplicacions → Programació → Geany*) o escrivint l'ordre *geany* des d'un terminal.

```
$ geany hola_c.c
```

A la finestra de l'editor s'ha d'escriure el codi font, següent:

```
#include <stdio.h>

extern void printHola();

int main() {
    printHola();
    printf("Hola!!!\n");
}
```

Un cop escrit el codi, deseu-lo indicant un nom amb l'estensió *.c*, per exemple: *hola_c.c*

4.4. Compilació del codi font C, assemblatge amb del codi objecte assemblador i generació de l'executable

La següent etapa consisteix a generar l'executable utilitzant el compilador *gcc*; per a fer-ho cal executar des d'un terminal (es pot utilitzar el terminal integrat amb l'editor Geany) l'ordre següent:

```
$ gcc -o hola_c -g hola_asm.o hola_c.c
```

Si el procés s'executa correctament, es generarà un nou fitxer, al mateix directori amb el nom indicat pel paràmetre *-o*; si no es posa aquest paràmetre, el fitxer executable generat s'anomena *a.out*.

4.5. Execució del programa

Per a executar el programa, cal escriure des del terminal (es pot utilitzar el terminal integrat amb l'editor Geany) el nom del executable, recordant d'indicar el directori actual al davant:

```
$ ./hola_c
```

El programa s'hauria d'executar correctament i mostrar els dos missatges “Hola!” i “Hola!!!”

```
$ ./hola_c
Hola!
Hola!!!
$
```

4.6. Depuració del programa amb KDbg

A continuació s'utilitzarà el depurador; cal executar des del terminal (es pot utilitzar el terminal integrat amb l'editor Geany) l'ordre següent:

```
$ kdbg hola_c
```

Un cop obert el depurador:

1) Definiu un punt de ruptura a la línia a la instrucció `printHola()`

2) Situeu-vos a la línia amb la instrucció `printHola()` i feu clic amb el ratolí sobre el signe + del principi de la línia; us apareixerà el codi assemblador per a aquella línia de codi C.

3) Inicieu l'execució del programa (*Run*) i executeu instrucció per instrucció, entrant dins les funcions:

Step into by instruction: Majúscules+F8 o *Execution → Step into by instruction*.

```

hola_c - KDbg
File Edit View Execution Breakpoint Settings Help
Open Source Find Reload Source Executable Run Step into Step over Step out Step into by instruction
hola_cc
+ 1 #include <stdio.h>
+ 2
+ 3 extern void printHola();
+ 4
+ 5 int main() {
+ 6     printHola();
+ 7     printf("Hola!!!\n");
+ 8
+ 9     return 0;
+10 }
+11

```

Quan entreu dins la subrutina `printHola` veureu el codi font del programa assemblador: `hola_asm.asm` (en una nova pestanya).

```

hola_c - KDbg
File Edit View Execution Breakpoint Settings Help
Open Source Find Reload Source Executable Run Step into Step over Step out Step into by instruction
hola_cc hola_asm.asm
+ 5 ;5:
+ 6 section .text ;6: Inici de la secció de codi
+ 7 global printHola ;7: Aquesta directiva és per fer visible
+ 8 ;8: una etiqueta per al compilador de C
+ 9 ;9:
+10 printHola: ;10: Nom que donem a la subrutina d'assemblador
+11 ;11: que cridarem des del programa en C
+12 ;12: Mostrar un missatge
+13 mov rax,4 ;13: Posa el valor 4 al registre rax per a fer
+14 ;14: la crida a la funció write (sys_write)
+15 mov rbx,1 ;15: Posa el valor 1 al registre rbx
+16 ;16: per indicar el descriptor que fa referència
+17 ;17: a la sortida estàndard
+18 mov rcx,msg ;18: Posa l'adreça de la variable msg
+19 ;19: al registre rcx
+20 mov rdx,6 ;20: Posa la longitud del missatge inclòs el 10
+21 ;21: del final al registre rdx
+22 int 80h ;22: crida al sistema operatiu
+23 ;23:
+24 ret ;24: retorn de la crida a subrutina.
+25 ;25:
+26

```

Podeu finalitzar l'execució instrucció a instrucció o tornar al codi C amb l'ordre *Execution → Step out* (F6).