

بخش چهارم

کسرا نوربخش 810100230

مقدمه

در این بخش با عملکرد بوت لودر آشنا شدیم و با یکی از آن‌ها یعنی GRUB، کار کردیم.

سوالات تشریحی

mekanizm-hayi GRUB

Grand Unified Boot Loader مسئول لود کردن یک کرنل و همچنین یک رم دیسک اولیه می‌باشد. به ما قابلیت‌های زیادی را همچون انتخاب از میان یکی از کرنل‌ها برای بوت شدن و operating mode برای بوت شدن را می‌دهد. در چندین مرحله (stage) کار می‌کند تا سازگار با محدودیت‌های سخت افزاری باشد. تنظیمات آن در grub.cfg موجود است که امکان تغییر مستقیم آن را نداریم (البته می‌توانیم اما توصیه نمی‌شود و همانطور که ما در ادامه عمل کردیم و بهتر است این است که تغییرات لازم را در /etc/default/grub داده و سپس با استفاده از دستور grub-update آن‌ها را اعمال کنیم).

GRUB یک منوی گرافیکی برای ما می‌آورد که امکان انتخاب کرنل‌های متفاوت موجود را داریم همچنین تنظیماتی از قبیل عکس background و ماجول‌هایی که نیاز است لود بشود و مدت زمانی که منتظر باشد تا به صورت اتومات، اولویت ما را بوت کند و غیره را در /etc/default/grub می‌توان تغییر داد.

UEFI & BIOS

Legacy Basic Input Output System در واقع جایی است که برای اولین بار سخت افزار با نرم افزار تعامل دارد و با استفاده از آن می‌توانیم نحوه Boot شدن سیستم را تغییر دهیم. BIOS با لود کردن MBR که 512 بایت ابتدایی بوت دیوایس انتخاب شده است، کار خود را شروع می‌کند و Bootstrap Code را که 440 بایت می‌باشد اجرا می‌کند که مرحله 1 بوت لودر است و به دنبال مرحله 2 بوت لودر در شروع یک پارتیشن می‌گردد. بایاس همچنین تست‌های اولیه یا همان POST را هم اجرا می‌کند.

مشکلی که در آن موجود است این است که حداقل 4 پارتیشن می‌توانیم داشته باشیم که این دست ما را می‌بندد از آن طرف

Unified Extensible Firmware Interface جایگرین بایاس می باشد که چیزی بیشتر از مشخصه هایی برای اینترفیس نیست و به پیاده سازی آن edk2 گویند. بجای استفاده از DPT(Disk Partition Table) که در بایاس بود از GPT(GUID Partition Table) استفاده می کند که این امکان را فراهم می آورد که تا 128 پارتیشن داشته باشیم و دیسک هایی با ظرفیت .8ZB

UEFI، FAT32 را می فهمد و با به کارگیری ESP که یک پارتیشن FAT32 است اجرا می شود. توانایی این را هم دارد که بدون لودر اجرا شود. همچنین با بهره گیری متداضماً و محاسبه checksum ها امکان secure boot را هم می دهد. همچنین قابلیت پشتیبانی از اینترنت را هم دارد.

بارگذاری و اجرای کرنل ها

بوت لودر این مسئولیت را دارد تا کرنل سیستم عامل مد نظر را مشخص و load کند. فرایند آن هم به این صورت است که بعد از مشخص شدن کرنل مد نظر، load می شود و در نهایت پروسه root که در سیستم های نوین لینوکسی Systemd می باشد شروع می شود.

ترتیب initialization یک کرنل به صورت: ابتدا یافتن CPU و Memory، جست و جو برای بس دیوایس و خود دیوایس، mount کردن روت فایل سیستم و ... این mount کردن یا همان بارگذاری به این صورت است که یک سری ماجول قابل لود شدن باید بارگذاری شوند اما چون شاید کرنل ما فایل سیستم mount شده ای را نداشته باشد باید این ماجول ها در جایی آرشیو شوند و بوت لودر این آرشیو را به مموری بیارد پیش از آن که کرنل اجرا شود.

وقتی که کرنل لود و initialize شد، وقت آن می رسد تا پردازه Systemd و پردازه های دیگر متوالی با آن در فضای کاربر ایجاد و اجرا شوند که مابقی فایل سیستم ها را mount می کنند و تمامی سرویس های شبکه را مقدار دهی اولیه می کنند و همچنین یک GUI نمایش داده می شود تا کاربر لایگین کند.

مفاهیم

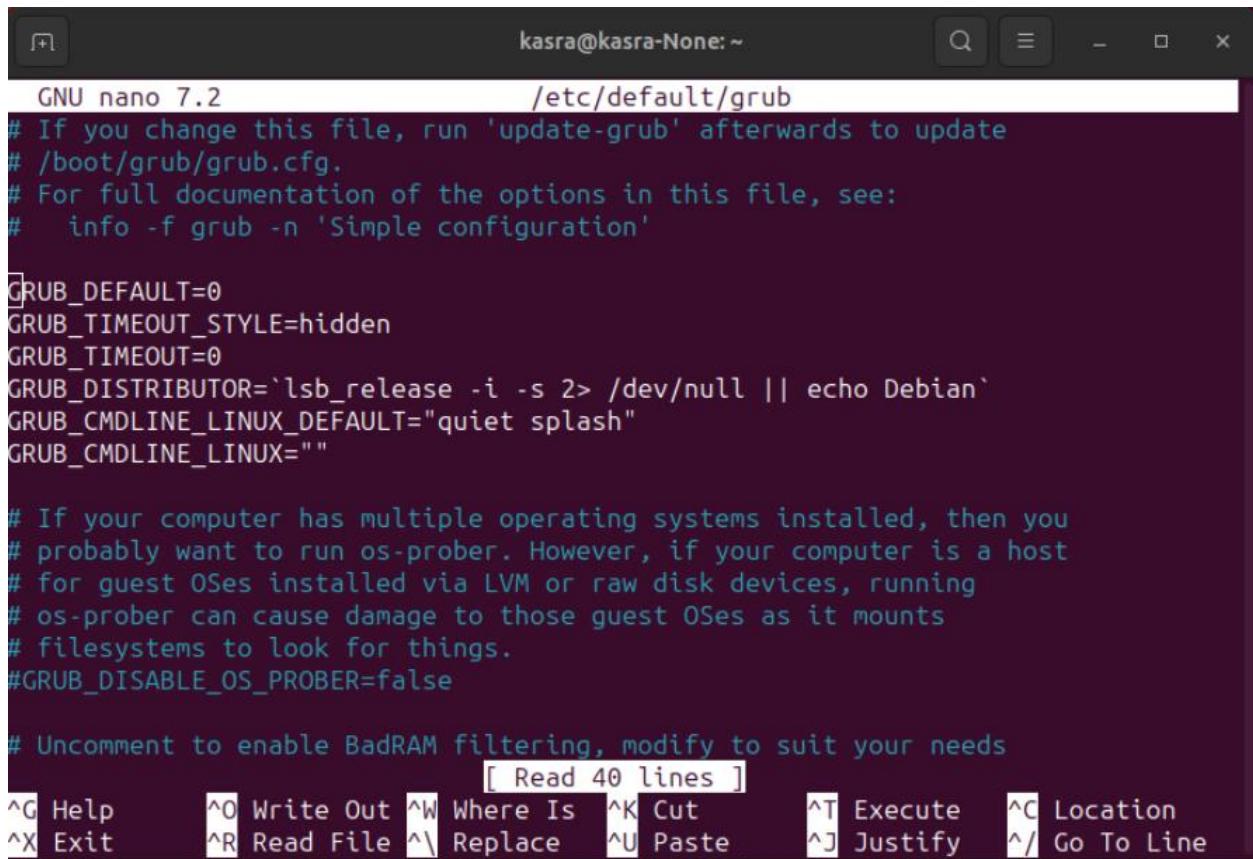
جدول پارتیشن: که در اینجا چون MBR را داریم، اطلاعات حداقل 4 پارتیشن را ذخیره می کند که این اطلاعات عبارت اند از موقعیت شروع و پایان هر پارتیشن، نوع پارتیشن و اکتیو بودن آن برای بوت شدن.

امضای بوت: 2 بایت پایانی MBR می باشد که در شکل کشیده شده AA 55 می باشد. که اگر با این مقدار تفاوتی وجود داشته باشد سیستم به احتمال زیاد اجازه بوت شدن نمی دهد و دیسک را شناسایی نمی کند.

برنامه **Winhex**: یک برنامه ادیتور هگز می باشد که با استفاده از آن می توانیم تحلیل و بازیابی داده ها را داشته باشیم. می توانیم با استفاده از آن محتوای MBR را مشاهده کرده و حتی تغییر دهیم، امضای بوت و جدول پارتیشن را بررسی کنیم و یا فایل های حذف شده و یا پارتیشن های خراب شده را بازیابی کنیم.

پیکربندی GRUB و مشاهده نتایج

من از ویرچوال ماشین استفاده کردم، داخل اپلیکیشن VMware Workstation و برای آن که منوی GRUB را ببینم، نیاز بود تا کلید شیفت سمت چپ را نگه دارم تا وارد منوی GRUB شود. این به دلیل آن بود که تنظیمات گراب من که با استفاده از sudo nano /etc/default/grub قابل مشاهده و تغییر است به این صورت بود:



```

GNU nano 7.2          /etc/default/grub
# If you change this file, run 'update-grub' afterwards to update
# /boot/grub/grub.cfg.
# For full documentation of the options in this file, see:
#   info -f grub -n 'Simple configuration'

GRUB_DEFAULT=0
GRUB_TIMEOUT_STYLE=hidden
GRUB_TIMEOUT=0
GRUB_DISTRIBUTOR=`lsb_release -i -s 2> /dev/null || echo Debian`
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash"
GRUB_CMDLINE_LINUX=""

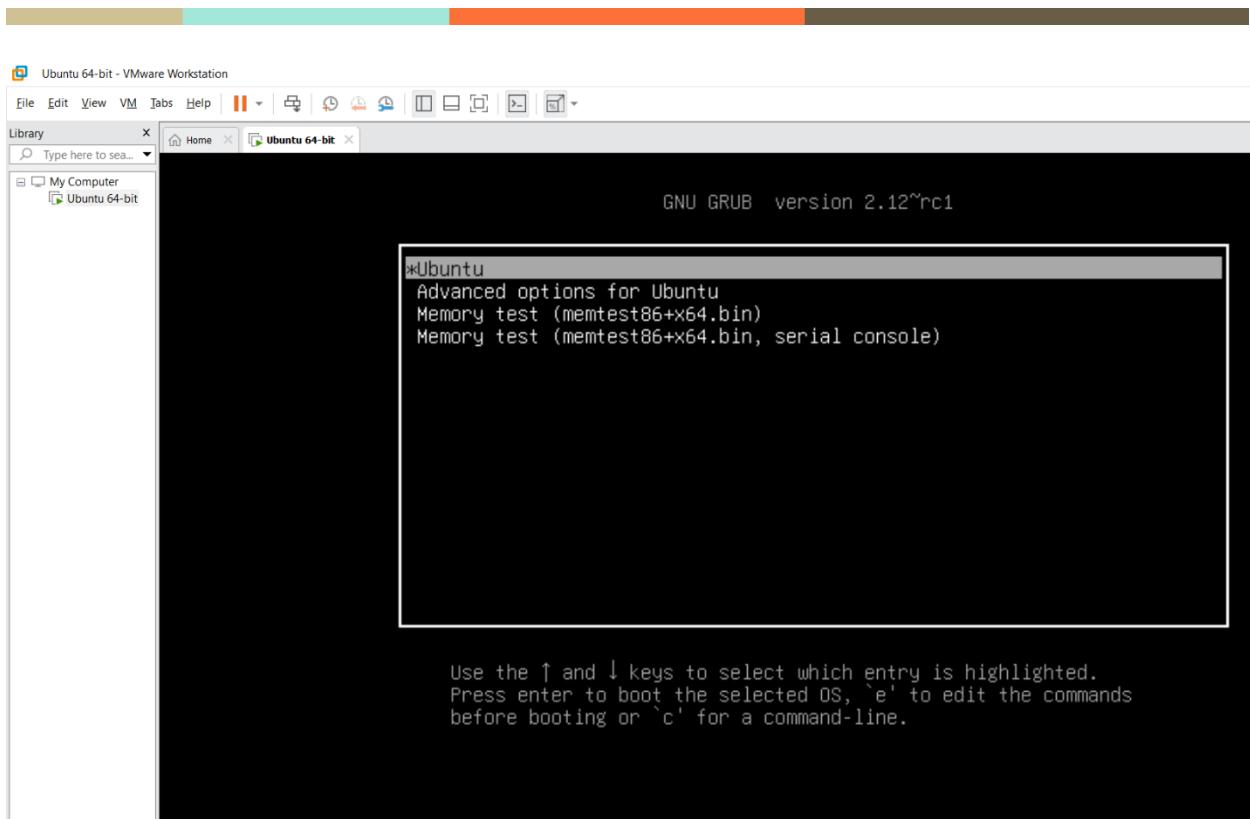
# If your computer has multiple operating systems installed, then you
# probably want to run os-prober. However, if your computer is a host
# for guest OSes installed via LVM or raw disk devices, running
# os-prober can cause damage to those guest OSes as it mounts
# filesystems to look for things.
#GRUB_DISABLE_OS_PROBER=false

# Uncomment to enable BadRAM filtering, modify to suit your needs
# [ Read 40 lines ]
^G Help      ^O Write Out  ^W Where Is  ^K Cut        ^T Execute    ^C Location
^X Exit      ^R Read File   ^\ Replace    ^U Paste     ^J Justify    ^/ Go To Line

```

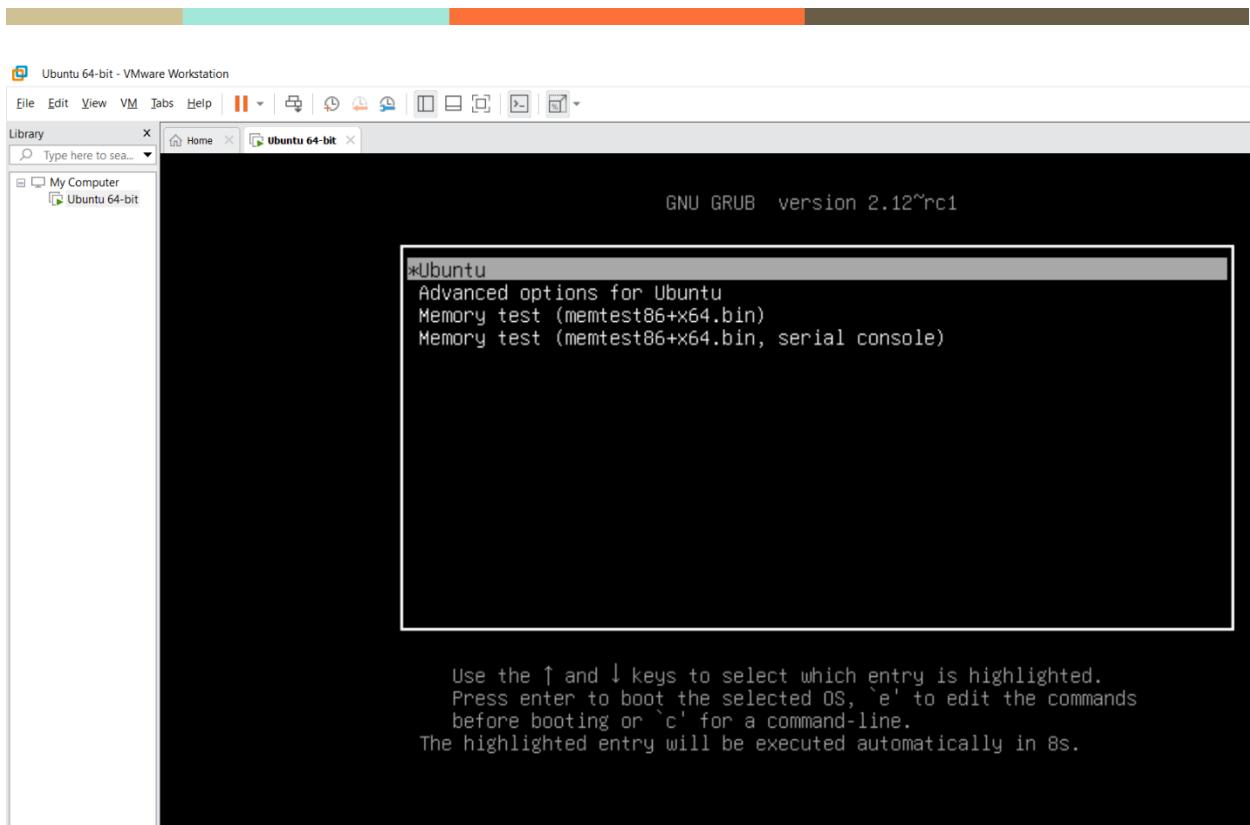
عکس ۱: تنظیمات GRUB

این یعنی اینکه کرنل دیفالت را اولین عنصر مشاهده شده در لیست قرار داده ایم، لیست انتخاب را مخفی کرده ایم و زمانی برای timeout نداریم و ۰ است یعنی درجا کرنل لود و اجرا می شود. (توضیح ۳ خط سفید اول در عکس بالا)



عکس 2: منوی گراب در حالت دیفالت که با شیفت وارد آن شدیم

در این بخش نیاز بود تا GRUB_TIMEOUT را به یک عددی تغییر داده تا به آن میزان منتظر باشد که آن را 10 گذاشتم و هم چنین GRUB_TIMEOUT_STYLE را از menu به hidden تغییر دهیم تا منوی GRUB نمایش داده شود. بعد از این تغییرات از دستور sudo update-grub استفاده کردیم تا این تغییرات ما بر روی grub.cfg هم اعمال شود و با یک بار ریبوت کردن سیستم نتیجه را مشاهده می کنیم که مستقیماً به منوی GRUB آمده ایم و نوشته است که بعد از 10 ثانیه به صورت اتومات، اوبونتو را اجرا می کند.



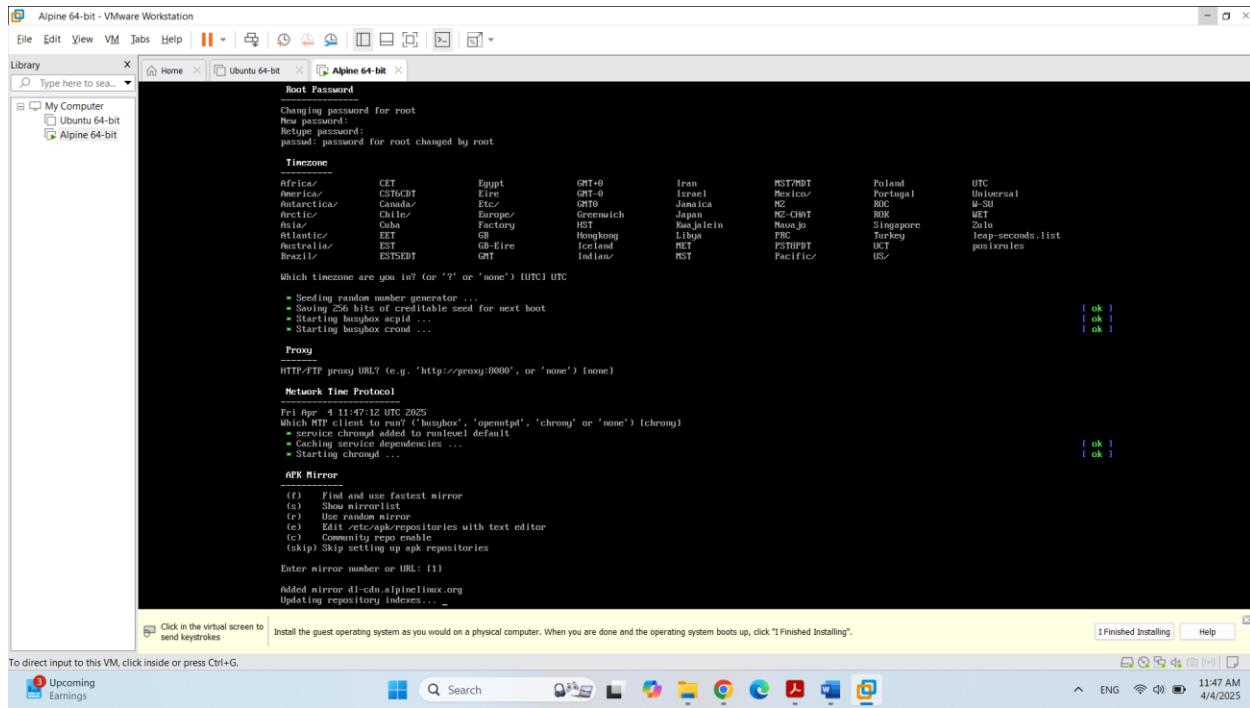
عکس ۳: نتیجه عوض کردن تنظیمات

با عوض کردن این مقدار، در واقع GRUB_TIMEOUT، آن را بر هر عددی که تنظیم کنیم، بعد از گذشت آن مقدار به صورت اتوماتیک سیستم بوت می شود.

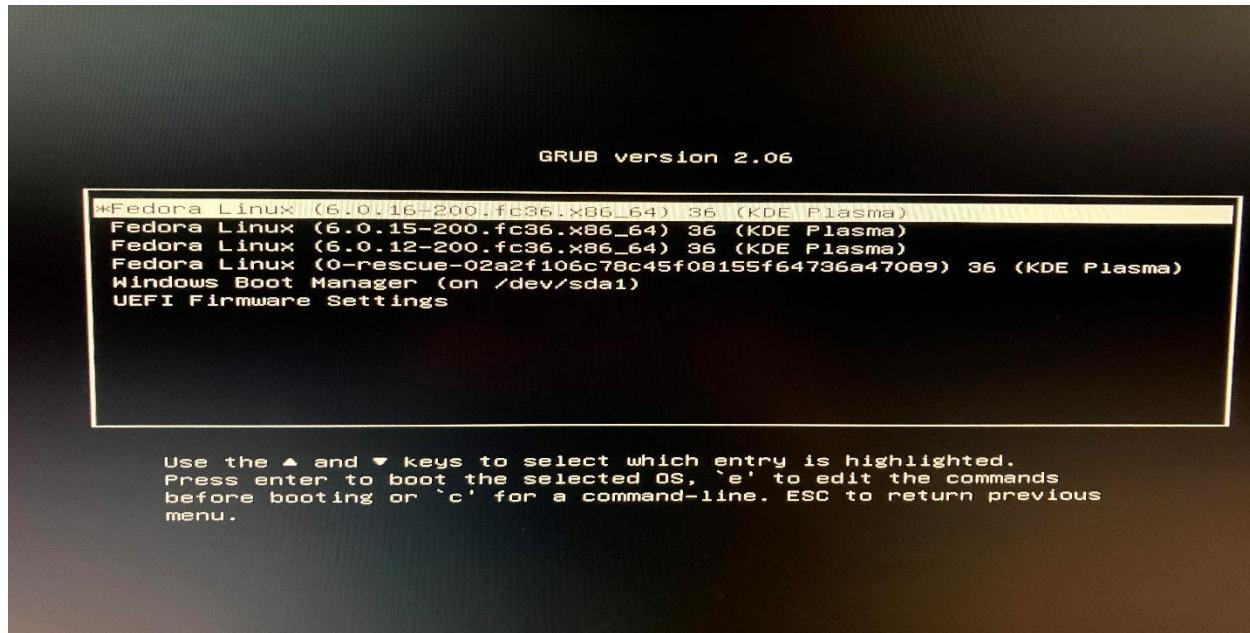
حال به سراغ نصب سیستم عامل دیگری رفتیم که در ابتدای Alpine استفاده کردیم و بعد از دانلود ISO آن به سراغ نصبش با استفاده از کامند setup-alpine و مقدار دهی به سوالات پرسیده شده رفتیم اما با خطا هایی نا ملموس مواجه شد به سراغ نصب کردن و اضافه کردن TinyCore به منوی بوت رفتیم اما آن را هم نتوانستیم نصب کنیم! من پیش تر در ترم های گذشته از GRUB استفاده کرده بودم که در آن می توانستم بین اوبونتو و یا ویندوز، انتخاب بکنم. مراحل اضافه کردن یک سیستم عامل به این صورت است که ابتدایاً آن را در ماشین مجازی اضافه کنیم و در نهایت که نصب شد، برای اینکه در منوی GRUB اضافه شود، یا باید آن را به عنوان یک disk، اضافه به VM ای که از پیش داشتیم بکنیم(از طریق VM) و یا این که آن را mount کنیم. حال که اوبونتو فرضاً آپاین را می شناسد، باید با استفاده از nano و یا هر ادیتور دیگر به /etc/grub.d/40_custom رفته یک خط یا به اصطلاح entry در منوی GRUB اضافه کنیم که به ترتیب زیر خواهد بود:

```
menuentry "Alpine Linux" {
    set root=(hdX, Y)
    linux /boot/vmlinuz-lts root=/dev/sdZ ro quiet
    initrd /boot/initramfs-lts }
```

که در آن (hdX, Y)، در واقع پارتیشن فرضاً GRUB ما می باشد که در Alpine notation نوشته شده است و روت پارتیشن سیستم عامل /dev/sdZ.



عکس ۴: مراحل نصب Alpine بعد از زدن دستور alpine setup



عکس ۵: نمونه منوی GRUB ای که در دستگاهی از پیش دیده بودم که سیستم عامل های مختلفی را دارد

حال ورودی ای را اضافه کردیم تا اطلاعات کلی سیستم را نشان دهد. برای این کار در مسیر `/opt/sysinfo.sh` یک اسکریپت نوشتیم که به صورت زیر می باشد و سپس با استفاده از `chmod +x` آن را قابل اجرا کردیم:

```
#!/bin/bash

echo "-----"
echo " SYSTEM INFORMATION"
echo "-----"
echo
".echo "Kernel and OS

(uname -a) استفاده از این می توانیم اطلاعات مربوط به سیستم عامل را ببینیم

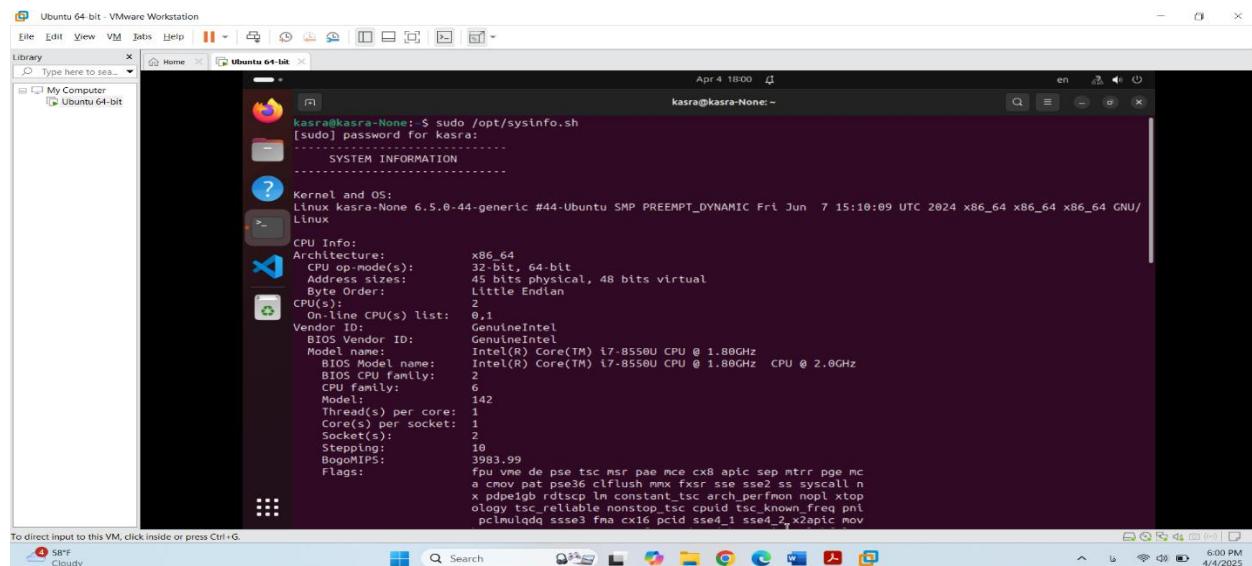
echo
".echo "CPU Info

lscpu
echo
".echo "Memory Info

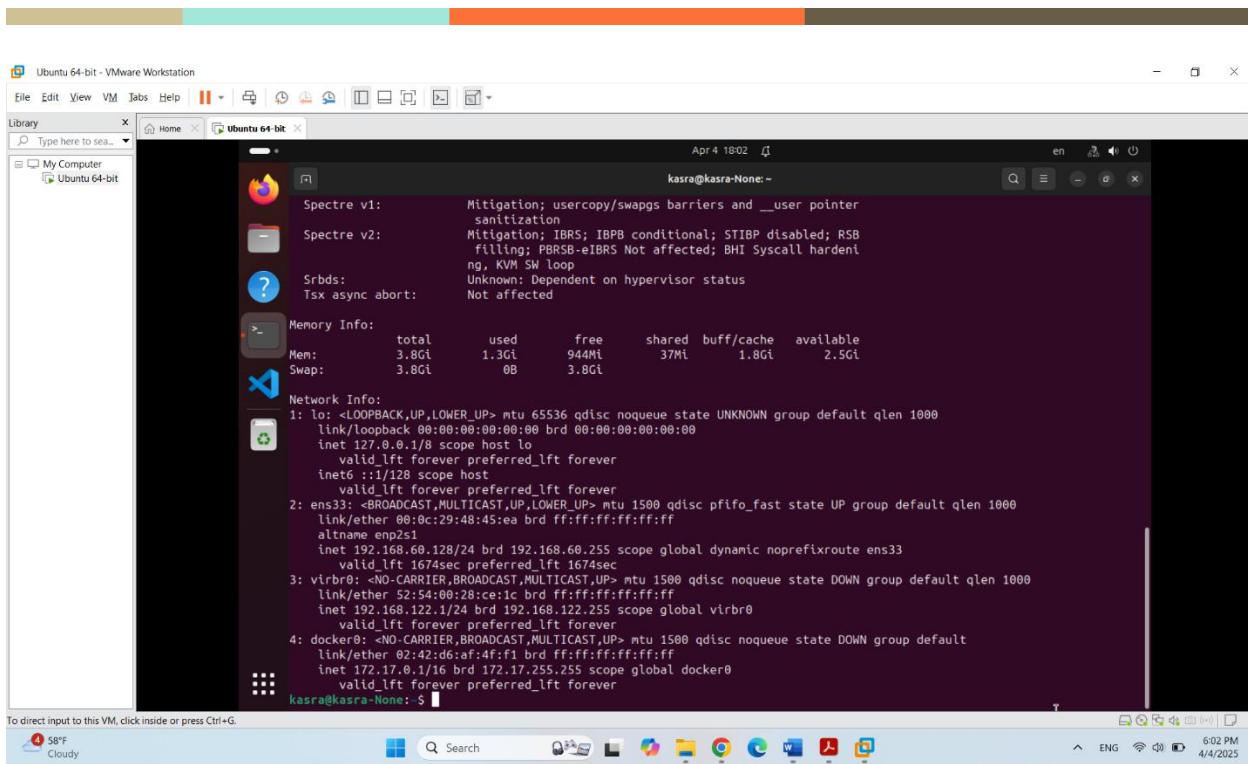
free -h
echo
".echo "Network Info

(ip a) اطلاعات مربوط به آیپی ادرس و اینترنت
```

برای اطمینان از درستی اسکریپت نوشته شده آن را با استفاده از دستور `sudo /opt/sysinfo.sh` آن را اجرا می کنیم:



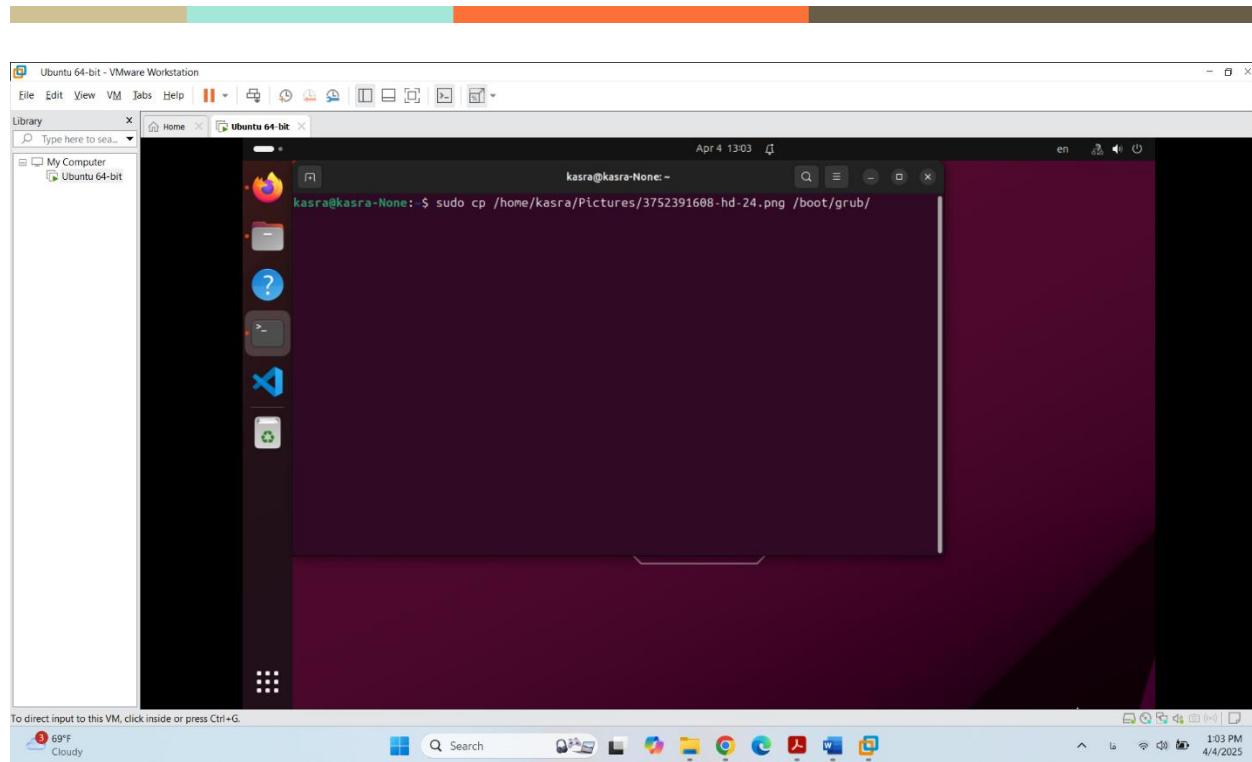
عکس 6: خروجی مربوط به بخش سیستم عامل



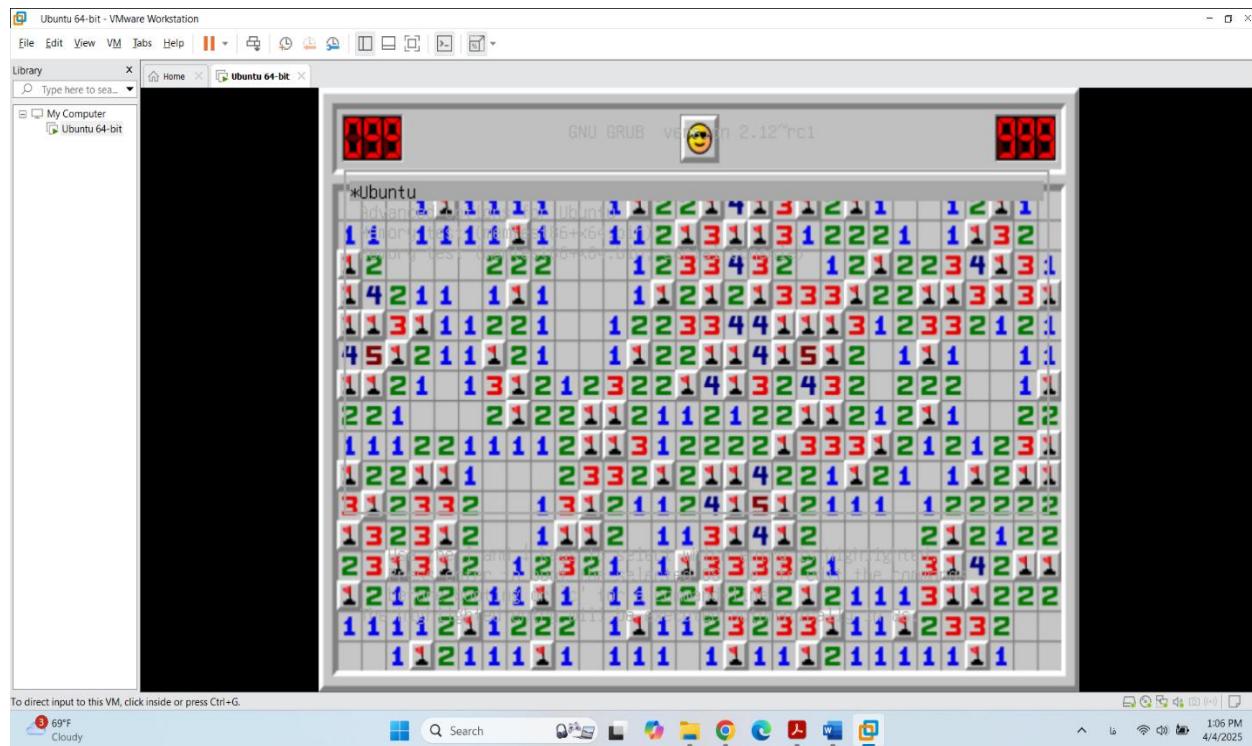
عکس 7: خروجی مربوط به بخش شبکه

سفارشی سازی GRUB

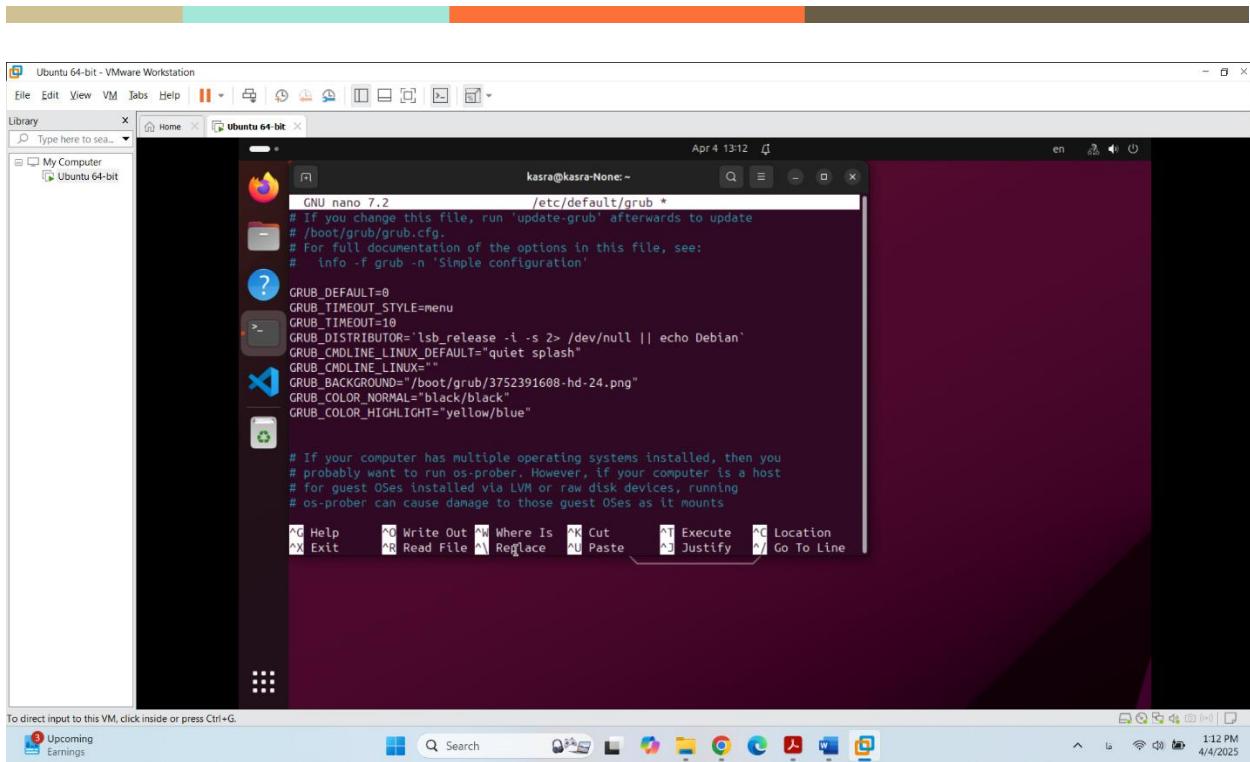
ابتدا دست به عوض کردن تصویر پس زمینه منو زدیم که باید ابتدا عکس‌مان را به آدرس /boot/grub/ کپی می‌کردیم و سپس با اضافه کردن خط "GRUB_BACKGROUND="/boot/grub/your_image.png" به تنظیمات /etc/default/grub در زدن، تنظیمات تغییر داده شده را اعمال می‌کنیم و در نهایت با پس زمینه جدید، منوی بوت را داریم. با update-grub تنظیمات مربوط به رنگ نوشته‌ها و همچنین هنگامی که یک منو آیتم سلکت می‌شود GRUB_COLOR_NORMAL و یا GRUB_COLOR_HIGHLIGHT را هم تغییر دهیم.



عکس 8: دستور کپی زده شده برای عکس پس زمینه



عکس ۹: منوی GRUB با پس زمینه جدید

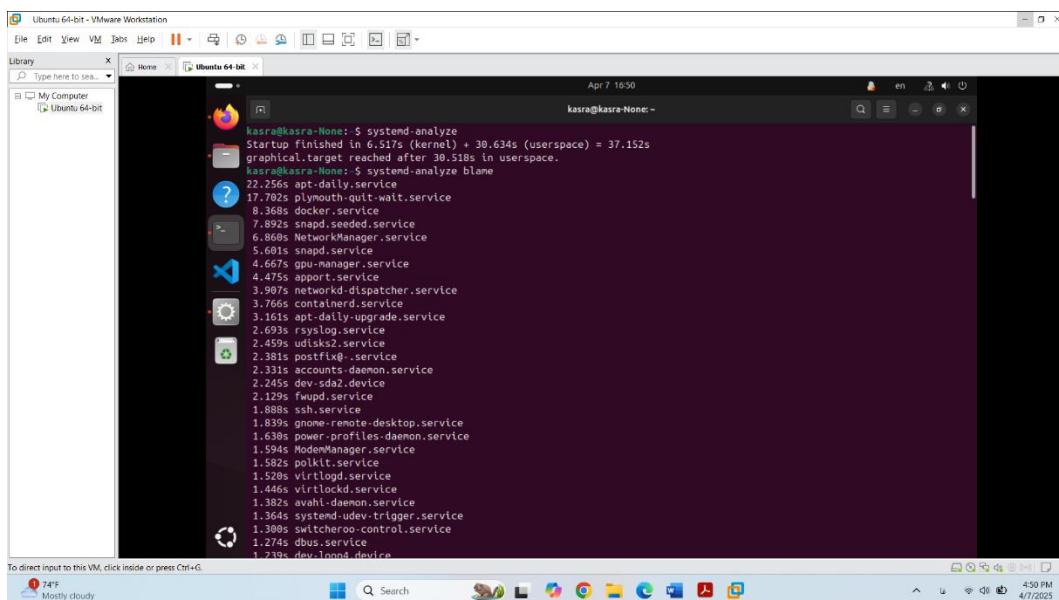


عکس 10: تنظیمات مورد نیاز برای تغییر دادن رنگ ها

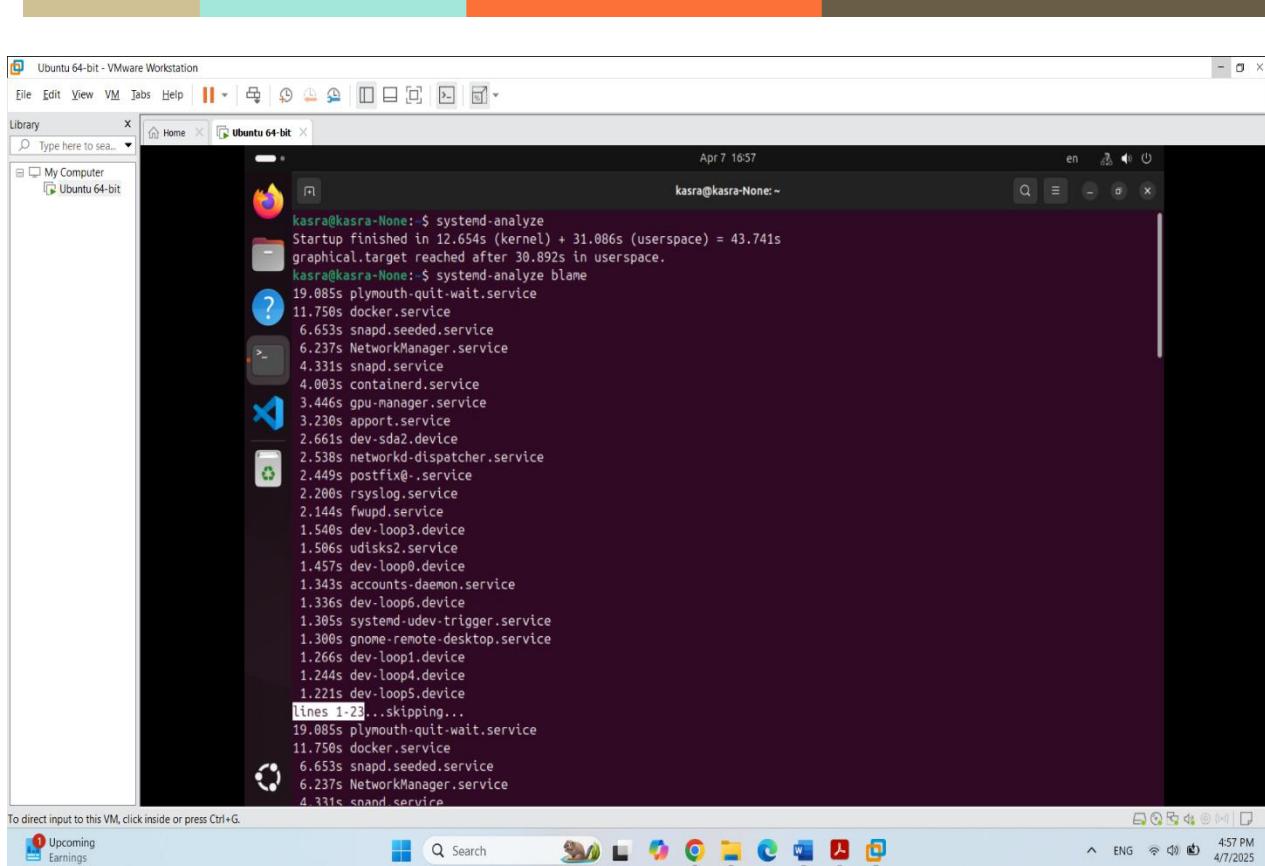
تحلیل عملکرد سیستم

در نهایت با استفاده از 2 دستور گفته شده یعنی، system-analyze blame و systemd-analyze خروجی ها را

گرفتیم:



عکس 11: خروجی 2 دستور قبل از تغییرات



عکس ۱۲: خروجی ۲ دستور بعد از تغییرات

لينک های استفاده شده در اين بخش

برای بخش سوالات تشریحی

https://en.wikipedia.org/wiki/Master_boot_record .1

<https://x-ways.net/winhex> .2

پیکربندی GRUB و مشاهده نتایج

<https://chatgpt.com/share/67f3cf70-8cc0-800d-abb1-05f6d7f67198> .1

استفاده از هوش مصنوعی ChatGPT

در بخشی که باید ما یک ورودی سفارشی در GRUB ایجاد می کردیم تا اسکریپت را اجرا کند، سوال را از آن به صورت:

I want to create a customized input in GRUB that runs a script when booting. A script that shows the system information. how can I achieve this on my ubuntu.

پرسیدم که چت کامل در لینک 1، در بالا موجود است.

چت کامل:

ChatGPT said:

Creating a custom GRUB boot menu entry that runs a script to show system information is possible but needs to be done carefully, as GRUB runs in a limited pre-boot environment. GRUB itself cannot run full Linux scripts — it can only load a kernel/initrd or run basic shell commands.

To achieve your goal, the usual approach is:

Create a custom GRUB menu entry.

Boot into a minimal Linux environment (like a custom initramfs).

Have that environment run your script showing system info.

Steps to Set This Up on Ubuntu

1. Create a Custom Script to Show System Info

Create a shell script somewhere (e.g., /opt/sysinfo.sh) that you want to run at boot:



```
bash
Copy
Edit
#!/bin/bash
echo "===== SYSTEM INFORMATION ====="
uname -a
lscpu
lsblk
free -h
ip a
Make it executable:
```

```
bash
Copy
Edit
sudo chmod +x /opt/sysinfo.sh
2. Create a Minimal Boot Environment
```

To run your script automatically at boot, you need to create a custom initramfs image that boots into a shell and runs your script.

Option 1: Use an existing minimal Linux (like Alpine) and customize it to run your script.

Option 2: Create your own initramfs.

Here's a basic way to create a custom initramfs:

```
bash
Copy
Edit
mkdir -p ~/initramfs/{bin,sbin,etc,proc,sys,newroot}
cd ~/initramfs
```

```
# Copy busybox (a single binary providing many common tools)
sudo apt install busybox
cp /bin/busybox bin/
```

```
# Create init script  
cat << 'EOF' > init  
#!/bin/sh  
mount -t proc none /proc  
mount -t sysfs none /sys
```

```
echo "Welcome to your custom boot info display"
```

```
sh /sysinfo.sh
```

```
exec sh
```

```
EOF
```

```
chmod +x init
```

```
# Copy your script  
cp /opt/sysinfo.sh sysinfo.sh  
chmod +x sysinfo.sh
```

```
# Create initramfs
```

```
find . | cpio -H newc -ov --owner root:root > ~/custom_initramfs.cpio.gz
```

3. Create a Custom GRUB Entry

Edit (or create) the GRUB custom entries file:

```
bash  
Copy  
Edit  
sudo nano /etc/grub.d/40_custom  
Add something like:
```

```
bash  
Copy  
Edit  
menuentry 'System Info Boot' {  
    insmod ext2
```



```
set root=(hd0,1) # Replace with your correct partition
linux /boot/vmlinuz-<version> root=/dev/ram0
initrd /boot/custom_initramfs.cpio.gz
}
```

Make sure to copy your `custom_initramfs.cpio.gz` to `/boot`.

To find your current kernel version:

```
bash
Copy
Edit
uname -r
Update GRUB:
```

```
bash
Copy
Edit
sudo update-grub
```

4. Test It

Reboot your machine and select “System Info Boot” from GRUB. It should launch the minimal environment and run your script showing system info.

Notes:

GRUB can't run full scripts. Your script must be executed after the kernel loads.

initramfs is a minimal root filesystem loaded into memory during boot. That's where your logic should go.

If you'd rather use something easier, consider using Alpine Linux or Tiny Core Linux, modifying their init to run your script.

Would you like a ready-to-use example with Alpine or want help building this from scratch?

You said:

how can I create the shell script in /opt/sysinfo.sh

ChatGPT said:

You can create the shell script at `/opt/sysinfo.sh` using `sudo` and a text editor like `nano` or `vim`. Here's a step-by-step:



Steps to Create /opt/sysinfo.sh

1. Open a terminal.
2. Use sudo to create and edit the script:

bash

Copy

Edit

sudo nano /opt/sysinfo.sh

3. Paste this content into the editor:

bash

Copy

Edit

#!/bin/bash

```
echo "====="
echo "  SYSTEM INFORMATION  "
echo "====="
echo
echo "Kernel and OS:"
uname -a
echo
echo "CPU Info:"
lscpu
echo
echo "Memory Info:"
free -h
echo
echo "Disk Info:"
lsblk
echo
echo "Network Info:"
ip a
```

4. Save and exit:

Press Ctrl + O to save.

Press Enter to confirm.

Press Ctrl + X to exit nano.

5. Make it executable:

bash

Copy

Edit

sudo chmod +x /opt/sysinfo.sh

Test It:

You can test the script right now by running:

bash

Copy

Edit

sudo /opt/sysinfo.sh

Let me know if you want to log the output to a file, run it in a GUI terminal, or customize the output further (like uptime, temperatures, or GPU info).

You said:

I should copy and paste all of the things you provided after this: Here's a basic way to create a custom initramfs:

in my terminal?

ChatGPT said:

Great question — you can't just copy and paste all of it in one go, because some parts are explanations or comments, and some require step-by-step interaction (like editing files). But I'll walk you through exactly what to copy-paste and when, so you can safely and successfully create your custom initramfs setup.

Step-by-Step: Create a Custom Initramfs

1. Create the directory structure

bash

Copy

Edit

mkdir -p ~/initramfs/{bin,sbin,etc,proc,sys,newroot}

cd ~/initramfs



2. Install and copy BusyBox

BusyBox provides minimal shell utilities for your initramfs.

bash

Copy

Edit

sudo apt update

sudo apt install busybox

cp /bin/busybox bin/

3. Create the init file

This script will be executed at boot time when the initramfs is loaded.

Create and open it:

bash

Copy

Edit

nano init

Paste this content into the editor:

sh

Copy

Edit

#!/bin/sh

mount -t proc none /proc

mount -t sysfs none /sys

echo "Welcome to your custom boot info display"

sh /sysinfo.sh

exec sh

Save and exit:

Ctrl + O → Enter

Ctrl + X

Make it executable:

bash
Copy
Edit
chmod +x init
4. Copy your sysinfo.sh script

You should've already created it in /opt/sysinfo.sh. Now copy it:

bash
Copy
Edit
sudo cp /opt/sysinfo.sh sysinfo.sh
chmod +x sysinfo.sh
5. Create the initramfs archive

Now package everything into a compressed initramfs file:

bash
Copy
Edit
find . | cpio -H newc -ov --owner root:root | gzip > ~/custom_initramfs.cpio.gz
This creates ~/custom_initramfs.cpio.gz.

Let me know once you reach this point, and I'll walk you through the GRUB custom entry setup and how to boot it!