PROJECT – NGUYÊN LÝ HỆ ĐỀU HÀNH

Học kỳ 1 – 2022 -2023

HỌ VÀ TÊN: TRẦN HOÀNG KIM MSSV: B2007245

Tuyên bố: Project này là do chính tôi, *Trần Hoàng Kim* (MSSV: *B2007245*), tự thực hiện, không sao chép của bất kỳ ai. Nếu có bất cứ sao chép nào, tôi hoàn toàn chịu trách nhiệm.

PHẦN 1: XÂY DỰNG LINUX KERNEL

Thực hiện ở chế độ người dùng root (root user)

$su-

A. CHUẨN BỊ LINUX KERNEL CODE (Ubuntu)

1. Dowload và cài đặt tool cần thiết vào hệ thống

$sudo apt-get install -y gcc libncurses5-dev make wget

```
kim@kim-VirtualBox:~$ sudo apt-get install -y gcc libncurses5-dev make wget
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
make is already the newest version (4.3-4.1build1).
wget is already the newest version (1.21.2-2ubuntu1).
wget set to manually installed.
The following additional packages will be installed:
  binutils binutils-common binutils-x86-64-linux-gnu gcc-11 gcc-12-base
  libasan6 libatomic1 libbinutils libc-dev-bin libc-devtools libc6-dev
  libcc1-0 libcrypt-dev libctf-nobfd0 libctf0 libgcc-11-dev libgcc-s1
  libgomp1 libitm1 liblsan0 libncurses-dev libnsl-dev libquadmath0 libstdc++6
  libtirpc-dev libtsan0 libubsan1 linux-libc-dev manpages-dev rpcsvc-proto
Suggested packages:
  binutils-doc gcc-multilib autoconf automake libtool flex bison gcc-doc
  gcc-11-multilib gcc-11-doc gcc-11-locales glibc-doc ncurses-doc
The following NEW packages will be installed:
  binutils binutils-common binutils-x86-64-linux-gnu gcc gcc-11 libasan6
  libatomic1 libbinutils libc-dev-bin libc-devtools libc6-dev libcc1-0
  libcrypt-dev libctf-nobfd0 libctf0 libgcc-11-dev libitm1 liblsan0
  libncurses-dev libncurses5-dev libnsl-dev libquadmath0 libtirpc-dev
  libtsan0 libubsan1 linux-libc-dev manpages-dev rpcsvc-proto
The following packages will be upgraded:
  gcc-12-base libgcc-s1 libgomp1 libstdc++6
4 upgraded, 28 newly installed, 0 to remove and 159 not upgraded.
Need to get 39.5 MB/40.4 MB of archives.
After this operation, 133 MB of additional disk space will be used.
Get:1 http://kh.archive.ubuntu.com/ubuntu jammy/main amd64 binutils-common amd6
4 2.38-3ubuntu1 [221 kB]
```

$sudo apt-get install -y gcc libssl-dev

```
kim@kim-VirtualBox:~$ sudo apt-get install -y gcc libssl-dev
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
gcc is already the newest version (4:11.2.0-1ubuntu1).
Suggested packages:
  libssl-doc
The following NEW packages will be installed:
  libssl-dev
0 upgraded, 1 newly installed, 0 to remove and 159 not upgraded.
Need to get 2,370 kB of archives.
After this operation, 12.4 MB of additional disk space will be used.
Get:1 http://kh.archive.ubuntu.com/ubuntu jammy-updates/main amd64 libssl-dev a
md64 3.0.2-0ubuntu1.6 [2,370 kB]
Fetched 2,370 kB in 4s (570 kB/s)
Selecting previously unselected package libssl-dev:amd64.
(Reading database ... 164398 files and directories currently installed.)
Preparing to unpack .../libssl-dev_3.0.2-0ubuntu1.6_amd64.deb ...
Unpacking libssl-dev:amd64 (3.0.2-0ubuntu1.6) ...
Setting up libssl-dev:amd64 (3.0.2-0ubuntu1.6) ...
kim@kim-VirtualBox:~$
```

$sudo apt-get install bison

```
kim@kim-VirtualBox:~$ sudo apt-get install bison
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libsigsegv2 m4
Suggested packages:
  bison-doc m4-doc
The following NEW packages will be installed:
  bison libsigsegv2 m4
0 upgraded, 3 newly installed, 0 to remove and 159 not upgraded.
Need to get 962 kB of archives.
After this operation, 2,924 kB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://kh.archive.ubuntu.com/ubuntu jammy/main amd64 libsigsegv2 amd64 2.
13-1ubuntu3 [14.6 kB]
Get:2 http://kh.archive.ubuntu.com/ubuntu jammy/main amd64 m4 amd64 1.4.18-5ubu
ntu2 [199 kB]
Get:3 http://kh.archive.ubuntu.com/ubuntu jammy/main amd64 bison amd64 2:3.8.2+
dfsg-1build1 [748 kB]
Fetched 962 kB in 3s (296 kB/s)
Selecting previously unselected package libsigsegv2:amd64.
(Reading database ... 164546 files and directories currently installed.)
Preparing to unpack .../libsigsegv2_2.13-1ubuntu3_amd64.deb ...
Unpacking libsigsegv2:amd64 (2.13-1ubuntu3) ...
Selecting previously unselected package m4.
Preparing to unpack .../m4_1.4.18-5ubuntu2_amd64.deb ...
Unpacking m4 (1.4.18-5ubuntu2) ...
Selecting previously unselected package bison.
```

$sudo apt-get install flex

```
kim@kim-VirtualBox:~$ sudo apt-get install flex
[sudo] password for kim:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libfl-dev libfl2
Suggested packages:
  build-essential flex-doc
The following NEW packages will be installed:
  flex libfl-dev libfl2
0 upgraded, 3 newly installed, 0 to remove and 159 not upgraded.
Need to get 324 kB of archives.
After this operation, 1,148 kB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://kh.archive.ubuntu.com/ubuntu jammy/main amd64 flex amd64 2.6.4-8bu
ild2 [307 kB]
Get:2 http://kh.archive.ubuntu.com/ubuntu jammy/main amd64 libfl2 amd64 2.6.4-8
build2 [10.7 kB]
Get:3 http://kh.archive.ubuntu.com/ubuntu jammy/main amd64 libfl-dev amd64 2.6.
4-8build2 [6,236 B]
Fetched 324 kB in 2s (139 kB/s)
Selecting previously unselected package flex.
(Reading database ... 164729 files and directories currently installed.)
Preparing to unpack .../flex_2.6.4-8build2_amd64.deb ...
Unpacking flex (2.6.4-8build2) ...
Selecting previously unselected package libfl2:amd64.
Preparing to unpack .../libfl2_2.6.4-8build2_amd64.deb ...
Unpacking libfl2:amd64 (2.6.4-8build2) ...
```

2. Xác định phiên bản hiện tại của kernel
   #uname -r

```
kim@kim-VirtualBox:~$ uname -r
5.15.0-43-generic
kim@kim-VirtualBox:~$
```

3. Dowload và giải nén

#**wget** https://www.kernel.org/pub/linux/kernel/v5.x/linux-5.15.48.tar.gz

```
kim@kim-VirtualBox:~$ wget https://cdn.kernel.org/pub/linux/kernel/v5.x/linux-5
.15.48.tar.gz
--2022-10-15 02:34:22--  https://cdn.kernel.org/pub/linux/kernel/v5.x/linux-5.1
5.48.tar.gz
Resolving cdn.kernel.org (cdn.kernel.org)... 151.101.1.176, 151.101.65.176, 151
.101.129.176, ...
Connecting to cdn.kernel.org (cdn.kernel.org)|151.101.1.176|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 195277562 (186M) [application/x-gzip]
Saving to: ‘linux-5.15.48.tar.gz’

linux-5.15.48.tar.g 100%[==================>] 186.23M  2.41MB/s    in 85s

2022-10-15 02:35:51 (2.20 MB/s) - ‘linux-5.15.48.tar.gz’ saved [195277562/19527
7562]
```

#tar xvzf linux-5.15.48.tar.gz

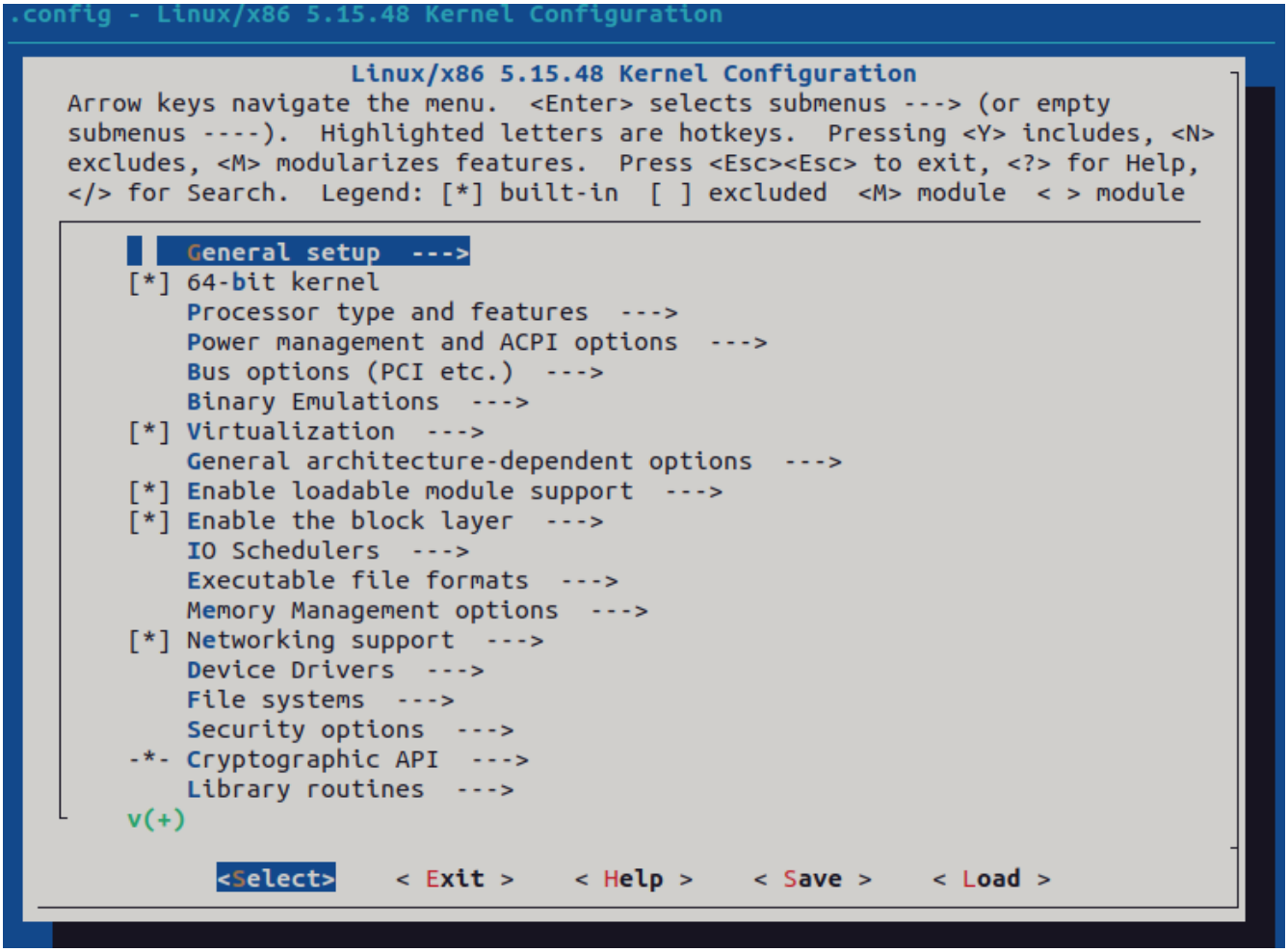
```
linux-5.15.48/usr/Makefile
linux-5.15.48/usr/default_cpio_list
linux-5.15.48/usr/gen_init_cpio.c
linux-5.15.48/usr/gen_initramfs.sh
linux-5.15.48/usr/include/
linux-5.15.48/usr/include/.gitignore
linux-5.15.48/usr/include/Makefile
linux-5.15.48/usr/initramfs_data.S
linux-5.15.48/virt/
linux-5.15.48/virt/Makefile
linux-5.15.48/virt/kvm/
linux-5.15.48/virt/kvm/Kconfig
linux-5.15.48/virt/kvm/async_pf.c
linux-5.15.48/virt/kvm/async_pf.h
linux-5.15.48/virt/kvm/binary_stats.c
linux-5.15.48/virt/kvm/coalesced_mmio.c
linux-5.15.48/virt/kvm/coalesced_mmio.h
linux-5.15.48/virt/kvm/dirty_ring.c
linux-5.15.48/virt/kvm/eventfd.c
linux-5.15.48/virt/kvm/irqchip.c
linux-5.15.48/virt/kvm/kvm_main.c
linux-5.15.48/virt/kvm/mmu_lock.h
linux-5.15.48/virt/kvm/vfio.c
linux-5.15.48/virt/kvm/vfio.h
linux-5.15.48/virt/lib/
linux-5.15.48/virt/lib/Kconfig
linux-5.15.48/virt/lib/Makefile
linux-5.15.48/virt/lib/irqbypass.c
kim@kim-VirtualBox:~$
```

B. CẤU HÌNH KERNEL MỚI

1. Đảm bảo đường dẫn hiện tại ở ~/**linux-5.15.48** và "**linux-5.15.48**" nằm ở top directory của kernel source.

```
kim@kim-VirtualBox:~/linux-5.15.48$
```

2. Tạo file cấu hình (config file)
   #make menuconfig



C. BIÊN DỊCH KERNEL

1- Tại **~/linux-5.15.48** tạo kernel giải nén (compressed kernel image)

#make –j4

Lỗi

```
make[1]: *** No rule to make target 'debian/canonical-certs.pem', needed by 'ce
rts/x509_certificate_list'.  Stop.
make[1]: *** Waiting for unfinished jobs....
  AS      arch/x86/purgatory/setup-x86_64.o
  CC      arch/x86/purgatory/sha256.o
  AR      arch/x86/net/built-in.a
  CC [M]  arch/x86/kvm/../../../virt/kvm/kvm_main.o
  AS      arch/x86/purgatory/entry64.o
  CC      arch/x86/purgatory/string.o
make: *** [Makefile:1852: certs] Error 2
make: *** Waiting for unfinished jobs....
```

Xử lý
Vào **#gedit .config** chỉnh sửa thông tin như hình

```
11132 CONFIG_SYSTEM_TRUSTED_KEYRING=y
11133 CONFIG_SYSTEM_TRUSTED_KEYS=""
11134 CONFIG_SYSTEM_EXTRA_CERTIFICATE=y
11135 CONFIG_SYSTEM_EXTRA_CERTIFICATE_SIZE=4096
11136 CONFIG_SECONDARY_TRUSTED_KEYRING=y
11137 CONFIG_SYSTEM_BLACKLIST_KEYRING=y
11138 CONFIG_SYSTEM_BLACKLIST_HASH_LIST=""
11139 CONFIG_SYSTEM_REVOCATION_LIST=y
11140 CONFIG_SYSTEM_REVOCATION_KEYS=""
```

Sau khi thực hiện xong

```
BTF [M] sound/usb/hiface/snd-usb-hiface.ko
BTF [M] sound/usb/caiaq/snd-usb-caiaq.ko
LD [M]  sound/usb/line6/snd-usb-line6.ko
LD [M]  sound/usb/line6/snd-usb-pod.ko
LD [M]  sound/usb/line6/snd-usb-podhd.ko
LD [M]  sound/usb/line6/snd-usb-toneport.ko
BTF [M] sound/usb/line6/snd-usb-pod.ko
BTF [M] sound/usb/line6/snd-usb-podhd.ko
BTF [M] sound/usb/line6/snd-usb-toneport.ko
BTF [M] sound/usb/line6/snd-usb-line6.ko
LD [M]  sound/usb/line6/snd-usb-variax.ko
LD [M]  sound/usb/misc/snd-ua101.ko
LD [M]  sound/usb/snd-usb-audio.ko
LD [M]  sound/usb/snd-usbmidi-lib.ko
BTF [M] sound/usb/line6/snd-usb-variax.ko
LD [M]  sound/usb/usx2y/snd-usb-us122l.ko
BTF [M] sound/usb/misc/snd-ua101.ko
BTF [M] sound/usb/usx2y/snd-usb-us122l.ko
BTF [M] sound/usb/snd-usbmidi-lib.ko
BTF [M] sound/usb/snd-usb-audio.ko
LD [M]  sound/usb/usx2y/snd-usb-usx2y.ko
LD [M]  sound/virtio/virtio_snd.ko
LD [M]  sound/x86/snd-hdmi-lpe-audio.ko
BTF [M] sound/x86/snd-hdmi-lpe-audio.ko
BTF [M] sound/usb/usx2y/snd-usb-usx2y.ko
BTF [M] sound/virtio/virtio_snd.ko
LD [M]  sound/xen/snd_xen_front.ko
BTF [M] sound/xen/snd_xen_front.ko
```

2- Biên dịch kernel modules

#make modules

```
  CC [M]  net/ipv4/netfilter/ip_tables.mod.o
  LD [M]  net/ipv4/netfilter/ip_tables.ko
  CC [M]  net/netfilter/x_tables.mod.o
  LD [M]  net/netfilter/x_tables.ko
  CC [M]  net/netfilter/xt_tcpudp.mod.o
  LD [M]  net/netfilter/xt_tcpudp.ko
  CC [M]  net/sched/sch_fq_codel.mod.o
  LD [M]  net/sched/sch_fq_codel.ko
  CC [M]  sound/ac97_bus.mod.o
  LD [M]  sound/ac97_bus.ko
  CC [M]  sound/core/seq/snd-seq.mod.o
  LD [M]  sound/core/seq/snd-seq.ko
  CC [M]  sound/core/snd-pcm.mod.o
  LD [M]  sound/core/snd-pcm.ko
  CC [M]  sound/core/snd-seq-device.mod.o
  LD [M]  sound/core/snd-seq-device.ko
  CC [M]  sound/core/snd-timer.mod.o
  LD [M]  sound/core/snd-timer.ko
  CC [M]  sound/core/snd.mod.o
  LD [M]  sound/core/snd.ko
  CC [M]  sound/pci/ac97/snd-ac97-codec.mod.o
  LD [M]  sound/pci/ac97/snd-ac97-codec.ko
  CC [M]  sound/pci/snd-intel8x0.mod.o
  LD [M]  sound/pci/snd-intel8x0.ko
  CC [M]  sound/soundcore.mod.o
  LD [M]  sound/soundcore.ko
  GEN     scripts/gdb/linux/constants.py
im@kim-VirtualBox:~/linux-5.15.48$
```

D. CÀI ĐẶT KERNEL

1- Cài đặt kernel modules

#make modules_install

```
SIGN    /lib/modules/5.15.48/kernel/lib/reed_solomon/reed_solomon.ko
INSTALL /lib/modules/5.15.48/kernel/net/ipv4/netfilter/ip_tables.ko
SIGN    /lib/modules/5.15.48/kernel/net/ipv4/netfilter/ip_tables.ko
INSTALL /lib/modules/5.15.48/kernel/net/netfilter/x_tables.ko
SIGN    /lib/modules/5.15.48/kernel/net/netfilter/x_tables.ko
INSTALL /lib/modules/5.15.48/kernel/net/netfilter/xt_tcpudp.ko
SIGN    /lib/modules/5.15.48/kernel/net/netfilter/xt_tcpudp.ko
INSTALL /lib/modules/5.15.48/kernel/net/sched/sch_fq_codel.ko
SIGN    /lib/modules/5.15.48/kernel/net/sched/sch_fq_codel.ko
INSTALL /lib/modules/5.15.48/kernel/sound/ac97_bus.ko
SIGN    /lib/modules/5.15.48/kernel/sound/ac97_bus.ko
INSTALL /lib/modules/5.15.48/kernel/sound/core/seq/snd-seq.ko
SIGN    /lib/modules/5.15.48/kernel/sound/core/seq/snd-seq.ko
INSTALL /lib/modules/5.15.48/kernel/sound/core/snd-pcm.ko
SIGN    /lib/modules/5.15.48/kernel/sound/core/snd-pcm.ko
INSTALL /lib/modules/5.15.48/kernel/sound/core/snd-seq-device.ko
SIGN    /lib/modules/5.15.48/kernel/sound/core/snd-seq-device.ko
INSTALL /lib/modules/5.15.48/kernel/sound/core/snd-timer.ko
SIGN    /lib/modules/5.15.48/kernel/sound/core/snd-timer.ko
INSTALL /lib/modules/5.15.48/kernel/sound/core/snd.ko
SIGN    /lib/modules/5.15.48/kernel/sound/core/snd.ko
INSTALL /lib/modules/5.15.48/kernel/sound/pci/ac97/snd-ac97-codec.ko
SIGN    /lib/modules/5.15.48/kernel/sound/pci/ac97/snd-ac97-codec.ko
INSTALL /lib/modules/5.15.48/kernel/sound/pci/snd-intel8x0.ko
SIGN    /lib/modules/5.15.48/kernel/sound/pci/snd-intel8x0.ko
INSTALL /lib/modules/5.15.48/kernel/sound/soundcore.ko
SIGN    /lib/modules/5.15.48/kernel/sound/soundcore.ko
DEPMOD  /lib/modules/5.15.48
```

1- Cài đặt kernel

#**sudo make install**

Lỗi

```
Warning: os-prober will not be executed to detect other bootable partitions.
Systems on them will not be added to the GRUB boot configuration.
Check GRUB_DISABLE_OS_PROBER documentation entry.
done
```

Xử lý

B1: #sudo apt install os-prober

```
kim@kim-VirtualBox:~/linux-5.15.48$ sudo apt install os-prober
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
os-prober is already the newest version (1.79ubuntu2).
0 upgraded, 0 newly installed, 0 to remove and 156 not upgraded.
kim@kim-VirtualBox:~/linux-5.15.48$ sudo nano /etc/default/grub
kim@kim-VirtualBox:~/linux-5.15.48$ sudo update-grub
Sourcing file `/etc/default/grub'
Sourcing file `/etc/default/grub.d/init-select.cfg'
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-5.15.48
Found initrd image: /boot/initrd.img-5.15.48
Found linux image: /boot/vmlinuz-5.15.0-43-generic
Found initrd image: /boot/initrd.img-5.15.0-43-generic
Found memtest86+ image: /boot/memtest86+.elf
Found memtest86+ image: /boot/memtest86+.bin
Warning: os-prober will be executed to detect other bootable partitions.
Its output will be used to detect bootable binaries on them and create new boot
 entries.
done
kim@kim-VirtualBox:~/linux-5.15.48$
```

B2: Gõ thêm dòng “GRUB_DISABLE_OS_PROBER=false” save và thoát.

```
  GNU nano 6.2                    /etc/default/grub *
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash"
GRUB_CMDLINE_LINUX=""

# Uncomment to enable BadRAM filtering, modify to suit your needs
# This works with Linux (no patch required) and with any kernel that obtains
# the memory map information from GRUB (GNU Mach, kernel of FreeBSD ...)
#GRUB_BADRAM="0x01234567,0xfefefefe,0x89abcdef,0xefefefef"

# Uncomment to disable graphical terminal (grub-pc only)
#GRUB_TERMINAL=console

# The resolution used on graphical terminal
# note that you can use only modes which your graphic card supports via VBE
# you can see them in real GRUB with the command `vbeinfo'
#GRUB_GFXMODE=640x480

# Uncomment if you don't want GRUB to pass "root=UUID=xxx" parameter to Linux
#GRUB_DISABLE_LINUX_UUID=true

# Uncomment to disable generation of recovery mode menu entries
#GRUB_DISABLE_RECOVERY="true"

# Uncomment to get a beep at grub start
#GRUB_INIT_TUNE="480 440 1"
GRUB_DISABLE_OS_PROBER=false

^G Help        ^O Write Out   ^W Where Is    ^K Cut         ^T Execute
^X Exit        ^R Read File   ^\ Replace     ^U Paste       ^J Justify
```

Kết quả

```
kim@kim-VirtualBox:~/linux-5.15.48$ sudo make install
arch/x86/Makefile:142: CONFIG_X86_X32 enabled but no binutils support
sh ./arch/x86/boot/install.sh 5.15.48 \
        arch/x86/boot/bzImage System.map "/boot"
run-parts: executing /etc/kernel/postinst.d/initramfs-tools 5.15.48 /boot/vmlin
uz-5.15.48
update-initramfs: Generating /boot/initrd.img-5.15.48
run-parts: executing /etc/kernel/postinst.d/unattended-upgrades 5.15.48 /boot/v
mlinuz-5.15.48
run-parts: executing /etc/kernel/postinst.d/update-notifier 5.15.48 /boot/vmlin
uz-5.15.48
run-parts: executing /etc/kernel/postinst.d/xx-update-initrd-links 5.15.48 /boo
t/vmlinuz-5.15.48
run-parts: executing /etc/kernel/postinst.d/zz-update-grub 5.15.48 /boot/vmlinu
z-5.15.48
Sourcing file `/etc/default/grub'
Sourcing file `/etc/default/grub.d/init-select.cfg'
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-5.15.48
Found initrd image: /boot/initrd.img-5.15.48
Found linux image: /boot/vmlinuz-5.15.48.old
Found initrd image: /boot/initrd.img-5.15.48
Found linux image: /boot/vmlinuz-5.15.0-43-generic
Found initrd image: /boot/initrd.img-5.15.0-43-generic
Found memtest86+ image: /boot/memtest86+.elf
Found memtest86+ image: /boot/memtest86+.bin
Warning: os-prober will be executed to detect other bootable partitions.
Its output will be used to detect bootable binaries on them and create new boot
```

E. THAY ĐỔI CẤU HÌNH GRUB (GRUB CONFIGURATION FILE)

Thay đổi cấu hình grub

#vim /etc/default/grub

Thực hiên thay đổi sau
GRUB_DEFAULT= 0
GRUB_TIMEOUT=25

```
 1 # If you change this file, run 'update-grub' afterwards to update
 2 # /boot/grub/grub.cfg.
 3 # For full documentation of the options in this file, see:
 4 #   info -f grub -n 'Simple configuration'
 5
 6 GRUB_DEFAULT=0
 7 GRUB_TIMEOUT_STYLE=hidden
 8 GRUB_TIMEOUT=25
 9 GRUB_DISTRIBUTOR=`lsb_release -i -s 2> /dev/null || echo Debian`
10 GRUB_CMDLINE_LINUX_DEFAULT="quiet splash"
11 GRUB_CMDLINE_LINUX=""
12
13 # Uncomment to enable BadRAM filtering, modify to suit your needs
14 # This works with Linux (no patch required) and with any kernel that obtains
15 # the memory map information from GRUB (GNU Mach, kernel of FreeBSD ...)
16 #GRUB_BADRAM="0x01234567,0xfefefefe,0x89abcdef,0xefefefef"
17
18 # Uncomment to disable graphical terminal (grub-pc only)
19 #GRUB_TERMINAL=console
20
21 # The resolution used on graphical terminal
22 # note that you can use only modes which your graphic card supports via VBE
23 # you can see them in real GRUB with the command `vbeinfo'
24 #GRUB_GFXMODE=640x480
```

F. REBOOT VM

1- Reboot kernel mới

#reboot

Khởi động lại máy ảo

2- Sau khi reboot kiểm tra thông tin kernel mới có đúng chưa

#**uname -r**

```
kim@kim-VirtualBox:~$ uname -r
5.15.48
kim@kim-VirtualBox:~$
```

PHẦN 2: THÊM LỜI GỌI HỆ THỐNG VÀO LINUX KERNEL

1- Download the kernel source code

#wget https://cdn.kernel.org/pub/linux/kernel/v6.x/linux-6.0.tar.gz

```
kim@kim-VirtualBox:~$ wget https://cdn.kernel.org/pub/linux/kernel/v6.x/linux-6
.0.tar.gz
--2022-10-18 06:47:45--  https://cdn.kernel.org/pub/linux/kernel/v6.x/linux-6.0
.tar.gz
Resolving cdn.kernel.org (cdn.kernel.org)... 151.101.1.176, 151.101.65.176, 151
.101.129.176, ...
Connecting to cdn.kernel.org (cdn.kernel.org)|151.101.1.176|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 214052479 (204M) [application/x-gzip]
Saving to: ‘linux-6.0.tar.gz’

linux-6.0.tar.gz    100%[===================>] 204.14M  4.21MB/s    in 80s

2022-10-18 06:49:11 (2.54 MB/s) - ‘linux-6.0.tar.gz’ saved [214052479/214052479
]

kim@kim-VirtualBox:~$
```

2- Extract the kernel source code

#tar xvzf linux-6.0.tar.gr

```
linux-6.0/virt/kvm/vfio.c
linux-6.0/virt/kvm/vfio.h
linux-6.0/virt/lib/
linux-6.0/virt/lib/Kconfig
linux-6.0/virt/lib/Makefile
linux-6.0/virt/lib/irqbypass.c
kim@kim-VirtualBox:~$
```

3- Define a new system call sys_helloworld()

- Create a dicetory helloworld in the kernel source directory :-

  #mkdir helloworld

  Change into this directory

  #cd helloworld

```
kim@kim-VirtualBox:~/linux-6.0$ mkdir helloworld
kim@kim-VirtualBox:~/linux-6.0$ cd helloworld
kim@kim-VirtualBox:~/linux-6.0/helloworld$
```

1. Create a “helloworld.c” file in this folder and add the definition of the system call to it as given below (you can use any text editor).

   #gedit helloworld.c

```
#include<linux/kernel.h>

asmlinkage long sys_helloworld(void)
{
    printk("Xin chao. Toi ten la Tran Hoang Kim\n");
    return 0;
}
```

2. Create a “Makefile” in the helloworld folder and add the given line to it.

   #gedit Makefile

   Add the following line to it :-

   Obj-y := hello.o

```
obj-y:=helloworld.o
```

- Add the helloworld directory to the kernel’s Makefile

  Change back into the linux-6.0 folder and open Makefile
  and add the helloworld at the end of core-y

#gedit Makefile

```
1102
1103 ifeq ($(KBUILD_EXTMOD),)
1104 core-y                  += kernel/ certs/ mm/ fs/ ipc/ security/ crypto/
     helloworld/
1105 core-$(CONFIG_BLOCK)    += block/
1106 core-$(CONFIG_IO_URING) += io_uring/
```

Add the new system call (sys_helloworld()) into the system call table (syscall_64.tbl file) (for 64 bit system)

#vi arch/x86/entry/syscalls/syscall_64.tbl

and add the end of line following

```
545     x32     execveat                compat_sys_execveat
546     x32     preadv2                 compat_sys_preadv64v2
547     x32     pwritev2                compat_sys_pwritev64v2
548     64      helloworld              sys_helloworld
# This is the end of the legacy x32 range.  Numbers 548 and above are
# not special and are not to be used for x32-specific syscalls.
-- INSERT --                                              418,55        Bo
```

Add the new system call (sys_helloworld()) in the system call header file.

#vi include/linux/syscalls.h

Add the following line to the end of the file just before #endif statement at the very bottom.

#asmlinkage long sys_helloworld(void);

```
                int optlen);
asmlinkage long sys_helloworld(void);
#endif
-- INSERT --                                            1388,38       Bo
```

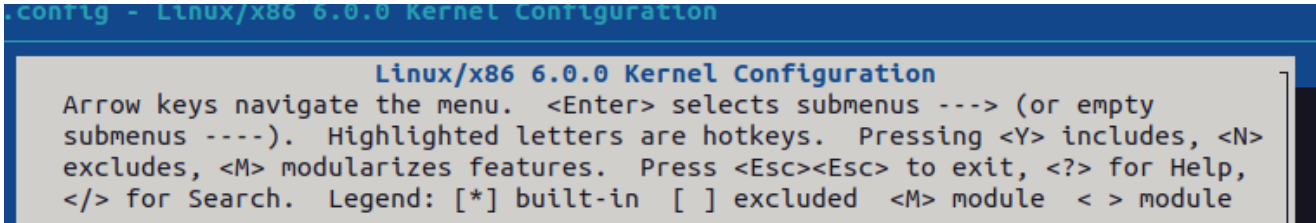
Compile kernel

1. Download necessary tools

2. Make menuconfig

#sudo make menuconfig

Save and exit



Now, compile the kernel (chỉnh sửa lỗi như phần trên)

#make

```
CC [M]  sound/usb/line6/snd-usb-toneport.mod.o
LD [M]  sound/usb/line6/snd-usb-toneport.ko
CC [M]  sound/usb/line6/snd-usb-variax.mod.o
LD [M]  sound/usb/line6/snd-usb-variax.ko
CC [M]  sound/usb/misc/snd-ua101.mod.o
LD [M]  sound/usb/misc/snd-ua101.ko
CC [M]  sound/usb/snd-usb-audio.mod.o
LD [M]  sound/usb/snd-usb-audio.ko
CC [M]  sound/usb/snd-usbmidi-lib.mod.o
LD [M]  sound/usb/snd-usbmidi-lib.ko
CC [M]  sound/usb/usx2y/snd-usb-us122l.mod.o
LD [M]  sound/usb/usx2y/snd-usb-us122l.ko
CC [M]  sound/usb/usx2y/snd-usb-usx2y.mod.o
LD [M]  sound/usb/usx2y/snd-usb-usx2y.ko
CC [M]  sound/x86/snd-hdmi-lpe-audio.mod.o
LD [M]  sound/x86/snd-hdmi-lpe-audio.ko
CC [M]  sound/xen/snd_xen_front.mod.o
LD [M]  sound/xen/snd_xen_front.ko
CC [M]  virt/lib/irqbypass.mod.o
LD [M]  virt/lib/irqbypass.ko
GEN     scripts/gdb/linux/constants.py
```

To install/ update the kernel

#sudo make modules_install

```
INSTALL sound/usb/hiface/snd-usb-hiface.ko
INSTALL sound/usb/line6/snd-usb-line6.ko
INSTALL sound/usb/line6/snd-usb-pod.ko
INSTALL sound/usb/line6/snd-usb-podhd.ko
INSTALL sound/usb/line6/snd-usb-toneport.ko
INSTALL sound/usb/line6/snd-usb-variax.ko
INSTALL sound/usb/misc/snd-ua101.ko
INSTALL sound/usb/snd-usb-audio.ko
INSTALL sound/usb/snd-usbmidi-lib.ko
INSTALL sound/usb/usx2y/snd-usb-us122l.ko
INSTALL sound/usb/usx2y/snd-usb-usx2y.ko
INSTALL sound/x86/snd-hdmi-lpe-audio.ko
INSTALL sound/xen/snd_xen_front.ko
INSTALL virt/lib/irqbypass.ko
DEPMOD  6.0
```

#sudo make install

```
sh ./arch/x86/boot/install.sh 6.0.0 arch/x86/boot/bzImage \
        System.map "/boot"
run-parts: executing /etc/kernel/postinst.d/apt-auto-removal 6.0.0 /boot/vmlinu
-6.0.0
run-parts: executing /etc/kernel/postinst.d/initramfs-tools 6.0.0 /boot/vmlinuz
6.0.0
update-initramfs: Generating /boot/initrd.img-6.0.0
run-parts: executing /etc/kernel/postinst.d/unattended-upgrades 6.0.0 /boot/vml
nuz-6.0.0
run-parts: executing /etc/kernel/postinst.d/update-notifier 6.0.0 /boot/vmlinuz
0.0
run-parts: executing /etc/kernel/postinst.d/zz-update-grub 6.0.0 /boot/vmlinuz-
.0.0
Sourcing file `/etc/default/grub'
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-6.0.0
Found initrd image: /boot/initrd.img-6.0.0
Found linux image: /boot/vmlinuz-5.15.48
Found initrd image: /boot/initrd.img-5.15.48
```

Reboot and test system call

#uname –r

```
kim@kim-VirtualBox:~/linux-6.0$ uname -r
6.0
```

Create “userspace.c”

```c
#include<linux/kernel.h>
#include<sys/syscall.h>
#include<stdio.h>
#include<unistd.h>

int main(){
      long int amma = syscall(548);
      printf("system call sys_helloworld return: %ld\n", amma);
      return 0;
}
```

Following these step
If return 0 is right

```
kim@kim-VirtualBox:~$ gcc userspace.c
kim@kim-VirtualBox:~$ ./a.out
system call sys_helloworld return: 0
kim@kim-VirtualBox:~$
```

#dmesg
This will display “Hello world” at the end of kernel’s message

```
[    20.472433] cryptd: max_cpu_qlen set to 1000
[    20.659018] AVX version of gcm_enc/dec engaged.
[    20.659020] AES CTR mode by8 optimization enabled
[    24.486854] snd_intel8x0 0000:00:05.0: white list rate for 1028:0177 is 48000
[    27.154762] Adding 4191228k swap on /dev/sda5.  Priority:-2 extents:1 across:
4191228k FS
[    40.013304] IPv6: ADDRCONF(NETDEV_UP): enp0s3: link is not ready
[    40.015584] IPv6: ADDRCONF(NETDEV_UP): enp0s3: link is not ready
[    40.020946] e1000: enp0s3 NIC Link is Up 1000 Mbps Full Duplex, Flow Control:
 RX
[    40.021268] IPv6: ADDRCONF(NETDEV_CHANGE): enp0s3: link becomes ready
[    53.655616] hrtimer: interrupt took 36892118 ns
[    60.758455] sched: RT throttling activated
[   864.557805] e1000: enp0s3 NIC Link is Down
[   866.574824] e1000: enp0s3 NIC Link is Up 1000 Mbps Full Duplex, Flow Control:
 RX
[   890.766081] e1000: enp0s3 NIC Link is Down
[   892.786106] e1000: enp0s3 NIC Link is Up 1000 Mbps Full Duplex, Flow Control:
 RX
[   894.798351] e1000: enp0s3 NIC Link is Down
[   896.815303] e1000: enp0s3 NIC Link is Up 1000 Mbps Full Duplex, Flow Control:
 RX
[ 5639.264688] helloworld
kim@kim-VirtualBox:~/linux-6.0$
```

PHẦN 3: ĐỊNH THỜI CPU

Thuật toán FCFS

- Đúng như cái tên của thuật toán Frist-Come First-Serve tiến trình nào tới đầu tiên sẽ được thực hiện trước.
- Ý tưởng: Ta sẽ dựa trên biểu đồ Grantt Chart và lưu trữu những thông tin trên biểu đồ đó bằng 1 cấu trúc mảng với mỗi tiến trình là một phần tử.

```
//Khai bao cau truc 1 process
typedef struct{
    int Name_Process; // 1->n
    int CPU_Time;
    int Arrival_Time;
    char State[LengthChart]; //chuoi dinh thoi
    int Waiting_Time;
    int Response_Time;
    int Turnarrond_Time;
    int head, tail;
}Process;
```

  - head: lưu trữ thời gian lần đầu tiên bắt gặp và thực hiện tiến trình.
  - tail: lưu trữ thời gian hoàn thành tiến trình.
  - Tùy theo mỗi bài toán non-premtive hay premtive head, tail sẽ có cấu trúc khác nhau.
- Khai báo 1 mảng cấu trúc các tiến trình.

```
//Khai bao cau truc danh sach tien trinh
typedef struct{
    Process process[NB_Process];
    int n;
}ListProcess;
```

- Các hàm hỗ trợ cho danh sách tiến trình

- void initListProcess(ListProcess1 *list): khởi tạo mảng tiến trình
- void appendListProcess(ListProcess1 *list, Process1 x): thêm 1 tiến trình vào mảng
- void Input(ListProcess *list): nhập dữ liệu
- void sort(ListProcess *list): sắp xếp tiến trình theo thời gian đến hệ thống
- void output (int Chart[], int index): in biểu đồ grantt chart

- Hàm FCFS

```
void FCFS(int Chart[], int *index, ListProcess *list, float *AVG_T, float *AVG_W, float *AVG_R
```

- Cần 1 mảng Chart và thật sự cần thiết 1 biến giữ tổng thời gian thực hiện hết tiến trình *index
- Khởi tạo các giá trị cần tính

```
*AVG_T = 0;
*AVG_W = 0;
*AVG_R = 0;
int i, check = 1, k;
*index = 0;
```

- Lặp trong n tiến trình và tính các giá trị head, tail của mỗi tiến trình.

```
for(i = 0; i < list->n; i++){
    while(check){
        if(list->process[i].Arrival_Time <= *index){
            list->process[i].head = *index;
            for(k = list->process[i].head; k <= list->process[i].CPU_Time+*index; k++){
                Chart[k] = list->process[i].Name_Process;
            }
            *index += list->process[i].CPU_Time;
            list->process[i].tail = *index;
            break;
        }
        else{
            Chart[*index] = 0;
            *index++;
        }
    }
}
```

- Kiểm tra tiến trình đó tới tại vị trí *index = 0 hay chưa không thì tăng *index++ và tại vị trí Chart[*index] này không có tiến trình nào nên gán bằng 0.
- Ngược lại nếu có, sẽ lặp CPU Time lần của tiến trình này tức là từ *index -> *index + CPU Time của tiến trình đó gán vào mảng Chart[*index] = tên tiến trình đại diện cho tại s đó.
- Tiến trình đó được thực hiện, head = *index và tail = *index + CPU Time, *index += CPU Time của tiến trình đó.

  - Xử lý các dự kiện cần tính.

```
//Xu ly cac du kien turnaround, waiting, response time
for(i = 0; i < list->n; i++){
    list->process[i].Turnaround_Time = list->process[i].tail - list->process[i].Arrival_Time;
    *AVG_T += list->process[i].Turnaround_Time;
    list->process[i].Response_Time = list->process[i].head - list->process[i].Arrival_Time;
    *AVG_R += list->process[i].Response_Time;
    list->process[i].Waiting_Time = list->process[i].head - list->process[i].Arrival_Time;
    *AVG_W += list->process[i].Waiting_Time;
}
*AVG_T /= list->n;
*AVG_R /= list->n;
*AVG_W /= list->n;
```

- Turnaround Time = Thời gian tiến trình đến hệ thống cho tời khi hoàn thành tiến
  trình.
- Respone Time = Thời gian tiến trình đến hệ thống cho tới khi bắt gặp tiến trình
  lần đầu tiên.
- Waiting Time = Thời gian tiến trình nằm trong hàng đợi Ready.
  - Ta dễ dàng tính được các giá trị trung bình.
  - Xử lý chuỗi định thời.
    - "R" đại diện cho tiến trình đang được thực hiện.
    - "W" đại diện cho tiến trình đang nằm trong hàng đợi Ready
    - "-" đại diện cho tiến trình đã thực hiện xong và không còn nằm trong hàng đợi Ready

```
//Xu ly du kien chuoi dinh thoi dua tren mang Chart[]
int j;
for(i = 0; i < list->n; i++){
    for(j = 0; j < *index; j++){ //State[0->index-1]
        if(j >= list->process[i].Arrival_Time && j < list->process[i].tail){
            if(list->process[i].Name_Process == Chart[j]){
                list->process[i].State[j] = 'R';
            }
            else list->process[i].State[j] = 'W';
        }
        else
        list->process[i].State[j] = '-';
    }
}
```

- Ta sẽ lặp trong tổng thời gian thực hiện các tiến trình tức là *index.
- Xét trong khoảng thời gian từ head => tail của mỗi tiến trình.
- Nếu tại đó tên tiến trình bằng với Chart[j] tức là tiến trình đang được thực hiện ta gán = "R"
- Ngược lại tức là tiến trình đang đợi trong hàng đợi Ready.
- Nếu vượt khỏi miền giá trị của tiến trình đó thì sẽ gán = "-" đã thực hiện xong.

  - Hàm main.

```
int chart[LengthChart], index, start;
float avgT, avgR, avgW;
ListProcess list;
Input(&list);
sort(&list);
start = list.process[0].Arrival_Time;
FCFS(chart, &index, &list, &avgT, &avgW, &avgR);
int i, j;
output(chart, index); // Grantt Chart
printf("\nDinh thoi FCFS\n");
for(i = 0; i < list.n; i++){
    for(j = start; j <= index; j++){
        printf("%c ", list.process[i].State[j]);
    }
    printf("\n");
}
printf("AVGW = %.2f\tAVGR = %.2f\tAVGT = %.2f",avgW, avgR, avgT);
printf("\n****************************************\n");
```

- Chạy chương trình

```
-----------------------------------MENU ALGORITHM-----------------------------------
                        1. FSFS algorithm
                        2. RounRobin algorithm
                        3. PriorityPremtive algorithm
                        0. exit


Choose the algorithm which you want: 1
Input number of process: 3
Input arrival time of the process 1 come system: 0
Input CPU time of the process 1: 2
Input arrival time of the process 2 come system: 1
Input CPU time of the process 2: 3
Input arrival time of the process 3 come system: 2
Input CPU time of the process 3: 2

Grantt Chart = [ 1 1 2 2 2 3 3 3 ]
Dinh thoi FCFS
R R - - - - -
- W R R R - -
- - W W W R R
AVGW = 1.33     AVGR = 1.33     AVGT = 3.67
********************************************
```

Thuật toán RoundRobin

- Thực hiện trong q thời gian phải trả CPU cho hệ thống
- Ý tưởng giống với thuật toán trên nhưng sẽ khác 1 ít trong phần cài đặt

```
//Khai bao cau truc pointX
typedef struct{
    int head, tail;
}pointX;

//Khai bao cau truc 1 process
typedef struct{
    int Name_Process; //1->n
    int CPU_Time;
    int Arrival_Time;
    int State[LengthChart];
    int Waiting_Time;
    int Respose_Time;
    int Turnaround_Time;
    pointX a[NB_Process];
    int sizePoint;
}Process1;
```

- Chú ý rằng ở đây không đơn thuần giữ 1 điểm head với 1 điểm tail bởi vì tiến trình chỉ được phép thực hiện trong q thời gian phải trả CPU về cho hệ thống nên ta phải giữ 1 mảng các head, tail bằng struct pointX, cho nên trong Struct

Process1 cần biến sizePoint để giữ số lượng các cặp head, tail này để tính các giá trị cần thiết.

- Khai báo struct 1 mảng các tiến trình

```
//Khai bao cau truc danh sach tien trinh
typedef struct{
    Process1 process[NB_Process];
    int n;
}ListProcess1;
```

- Các hàm hỗ trợ cho danh sách tiến trình
  - void initListProcess1(ListProcess1 *list): khởi tạo mảng tiến trình
  - void appendListProcess1(ListProcess1 *list, Process1 x): thêm 1 tiến trình vào mảng
  - void Input1(ListProcess *list): nhập dữ liệu
  - void sort1(ListProcess *list): sắp xếp tiến trình theo thời gian đến hệ thống
  - void output1(int Chart[], int index): in biểu đồ grantt chart

- Ta sẽ khai báo thêm 1 struct hàng đợi để giữ các tiến trình thực hiện chưa xong CPU Time

```
//Khai bao cau truc hang doi
typedef struct{
    Process1 Elements[MaxLength];
    int Front, Rear;
}Queue;
```

- Các hàm hỗ trợ cho hàng đợi Queue
  - void makeNullQueue(Queue *Q): khởi tạo hàng đợi
  - int isEmptyQueue(Queue Q): kiểm tra hàng đợi rỗng ?
  - int isFullQueue(Queue Q): kiểm tra hàng đợi đầy ?
  - void appendQueue(Queue *Q, Process1 x): thêm 1 phần tử vào hàng đợi
  - Process1 Front(Queue Q): trả về tiến trình đầu hàng đợi
  - int headToPostion(Process1 p, ListProcess1 list): Xác định vị trí tiến trình trong
    danh sách các tiến trình
  - void deQueue(Queue *Q): xóa phần tử trong hàng đợi
  - int equals(Process1 x, Process1 y): trả về true nếu x.name == y.name else false
  - int member(Queue Q, Process1 x): hàm thành viên kiểm tra xem 1 tiến trình nào đó
  - có nằm trong hàng đợi hay không
- Hàm RR

```
void RR(int Chart[], int *index, ListProcess1 *list, float *AVG_T, float *AVG_W, float *AVG_R, int q, int *start)
```

- Với *index sẽ giữ tổng thời gian của tiến trình và *start sẽ xác định giây mà tiến trình tới đầu tiên nhất

  - Khởi tạo các giá trị cần thiết

```
*AVG_T = 0;
*AVG_W = 0;
*AVG_R = 0;
Queue Q; makeNullQueue(&Q);
int count = 0, k, i, check = 1, j, total_Time = 0;
```

- Khai báo và khởi tạo thêm 1 hàng đợi Q

- Công việc đầu tiên là xác định tiến trình đầu tiên sẽ được thêm vào hàng đợi

```
int min_Arrival_Time = INF, position;
for(i = 0; i < list->n; i++){
    if(list->process[i].Arrival_Time <= min_Arrival_Time){
        min_Arrival_Time = list->process[i].Arrival_Time;
        position = i;
    }
}
//Them tien trinh vao hang doi va *index bat dau tu vi tri nay
appendQueue(&Q, list->process[position]);
*start = list->process[position].Arrival_Time;
*index = list->process[position].Arrival_Time;
```

  - Nếu tiến trình nào có Arrival_Time nhỏ nhất sẽ được thêm vào hàng đợi Và tính giá trị cho *start, *index
  - Khởi tạo mảng head, tail cho từng tiến trình

```
//Khoi tao sizePoint
for(i = 0; i < list->n; i++){
    list->process[i].sizePoint = 0;
}
```

  - Vòng lặp chính

```
//Lap cho toi khi hang doi rong
while(!isEmptyQueue(Q)){
    Process1 p1 = Front(Q);
    deQueue(&Q);
    int pos = headToPostion(p1, *list), cpu_run;
    //Head
    list->process[pos].a[list->process[pos].sizePoint].head = *index;
    //Head
    cpu_run = list->process[pos].CPU_Time;
    for(k = 0; k <= list->process[pos].CPU_Time+*index; k++){
        if(count == q || cpu_run == 0){

            //Them process ke tiep vao hang doi voi dk s do toi va ko nam trong hang doi
            for(i = 0; i < list->n; i++){
                while(check){
                    //giay do toi/khong nam trong Queue/khac voi tien trinh dang xet
                    if(list->process[i].CPU_Time != done && list->process[i].Arrival_Time <= *index &&
                    !member(Q, list->process[i]) && list->process[i].Name_Process != list->process[pos].Name_Process
                        appendQueue(&Q, list->process[i]);
                    }
                    break;
```

```
            }
        }

        //Giam cpu time va neu cpu time > 0 thi them vao cuoi hang doi
        list->process[pos].CPU_Time -= q;
        if(list->process[pos].CPU_Time > 0){
            appendQueue(&Q, list->process[pos]);
            list->process[pos].a[list->process[pos].sizePoint].tail = *index;
            //Tang sizePoint
            list->process[pos].sizePoint++;
        }
        else{
            list->process[pos].a[list->process[pos].sizePoint].tail = *index;
            list->process[pos].CPU_Time = done;
        }
        total_Time += list->process[pos].CPU_Time;
        count = 0;
        break;
    }

    else{
        //danh dau tren mang Chart
        Chart[*index] = list->process[pos].Name_Process;
        count++;
        *index += 1;
        cpu_run--;
    }
  }
}
```

- Đầu tiên phần tử đâu tiên được lấy ra và xóa khỏi trong hàng đợi
- Biến pos dùng để xác định xem tiến trình trong hàng đợi đang ở index nào trong ListProcess
- Biến cpu_run dùng để đếm số lần q
- Tính giá trị head trong sizePoint đã được cấp phát ở trên
- Gán cpu_run = CPU Time của tiến trình (CPU Time của tiến trình sẽ đc cập nhật sau mỗi lần lặp)
- Lặp trong CPU Time của tiến trình, ở đây bắt đầu từ k = 0 cũng không ảnh hưởng nhiều đến chương trình vì ngay ở bên dưới có 1 lệnh if để check điều kiện và break bất cứ lúc nào nếu điều kiện đúng
- Với lệnh if(count == q || cpu_run == 0) nếu không đúng điều kiện thì sẽ chuyển xuống lệnh else bên dưới, mảng Chart[*index] sẽ được gán tên tiến trình, count++ đến số q tăng lên và cpu_run-- cập nhật lại cpu_run để tiếp tục check điều kiện dừng
- Nếu đúng với điều kiện đó chúng ta sẽ tiến hành tìm các tiến trình đến trong giây đó với điều kiện “Tiến trình đó chưa xong tức là != done + Thời gian đến <= *index + chưa nằm trong hàng đợi đang lặp + tên tiến trình đó phải khác tiến trình đang xét’
- Cập nhật lại các giá trị CPU Time

        - if CPU Time > 0 tức là vẫn chưa thực hiện xong, ta sẽ tính giá trị tail cho nó và phải tăng sizePoint để giữ head, tail tiếp theo
        - Ngược lại thì ta chỉ tính tail cho nó và gán CPU Time == done đánh dấu hoàn thành tiến trình
    - Biến total_time giữ số giây thực hiện các tiến trình sẽ được cập nhật sau mỗi lần lặp
    - Reset biến count để đếm số lần q về = 0
    - Break
    - Tiếp tục lặp cho tới khi hoàn thành xong hết các tiến trình

- Tính các giá trị cần thiết

```
//Tinh cac gia tri can thiet (Chu y: phai la chi so index + 1 de lay s chinh xac)
for(i = 0; i < list->n; i++){
    *AVG_T += list->process[i].a[list->process[i].sizePoint].tail - list->process[i].Arrival_Time;
    *AVG_R += list->process[i].a[0].head - list->process[i].Arrival_Time;
    for(j = 0; j <= list->process[i].sizePoint; j++){
        if(j == 0){
            *AVG_W += list->process[i].a[0].head - list->process[i].Arrival_Time;
        }
        else{
            *AVG_W += list->process[i].a[j].head - list->process[i].a[j-1].tail;
        }
    }
}
*AVG_T /= list->n;
*AVG_R /= list->n;
*AVG_W /= list->n;
```

Ta thấy rằng tại vị trí sizePoint tiến trình mới thực hiện xong
Tại vị trí sizePoint = 0 thì đó là lần đầu tiên đến hệ thống
Như đã nói ở trên thuật toán này sẽ giữ mảng head, tail để giữ đầu cuối của mỗi Running Time
Nên ở đây ta cần tạo 1 vòng lặp để tính nó
Ta dễ dàng tính được các giá trị trên

- Xử lý trên chuỗi định thời

```
//Xu ly du kien chuoi dinh thoi tren mang Chart[]
int z;
for(i = 0; i < list->n; i++){
    for(j = 0; j < *index; j++){
        if( j >= list->process[i].Arrival_Time && j < list->process[i].a[list->process[i].sizePoint].tail){
            if(list->process[i].Name_Process == Chart[j]){
                list->process[i].State[j] = 'R';
            }
            else list->process[i].State[j] = 'W';
        }
        else
        list->process[i].State[j] = '-';
    }
}
```

- Cũng tương tự như bài toán FSFC nhưng ta phải cần vòng lặp lồng cho mảng head, tail để xét từng miền giá trị cho mỗi cặp giá trị đó
- Hàm main

```
int chart[LengthChart], index, q, i, j, start;
float avgT, avgR, avgW;
ListProcess1 list;
Input1(&list, &q);
sort1(&list);
RR(chart, &index, &list, &avgT, &avgW, &avgR, q, &start);
printf("Grantt Chart = [ ");
for(i = 0; i < index; i++){
    printf("%d ", chart[i]);
}
printf("]");
printf("\nDinh Thoi RoundRobin\n");
for(i = 0; i < list.n; i++){
    for(j = start; j <= index; j++){
        printf("%c ", list.process[i].State[j]);
    }
    printf("\n");
}
printf("AVGW = %.2f\tAVGR = %.2f\tAVGT = %.2f",avgW, avgR, avgT);
printf("\n****************************************\n");
```

- Chạy chương trình

```
-----------------------------------MENU ALGORITHM-----------------------------------
                        1. FSFS algorithm
                        2. RounRobin algorithm
                        3. PriorityPremtive algorithm
                        0. exit


Choose the algorithm which you want: 2
Input quantum q = 3
Input number of process n = 6
Input arrival time of the process 1 come system: 0
Input CPU time of the process 1: 4
Input arrival time of the process 2 come system: 1
Input CPU time of the process 2: 5
Input arrival time of the process 3 come system: 2
Input CPU time of the process 3: 2
Input arrival time of the process 4 come system: 3
Input CPU time of the process 4: 1
Input arrival time of the process 5 come system: 4
Input CPU time of the process 5: 6
Input arrival time of the process 6 come system: 6
Input CPU time of the process 6: 3
Grantt Chart = [ 1 1 1 2 2 2 3 3 4 1 5 5 5 6 6 6 2 2 5 5 5 ]
Dinh Thoi RoundRobin
R R R W W W W W W R - - - - - - - - - - -
- W W R R R W W W W W W W W W R R - - -
- - W W W W R R - - - - - - - - - - - -
- - - W W W W W R - - - - - - - - - - -
- - - - W W W W W W R R R W W W W W R R R
- - - - - - W W W W W W W R R R - - - - -
AVGW = 7.50     AVGR = 4.00     AVGT = 11.00
****************************************
```

Thuật toán Priority Premtive

- Tiến trình nào được thực hiện trước phụ thuộc vào thời gian đến và piority của tiến trình đó
- Ý tưởng cũng tương tự như 2 thuật toán ở trên nhưng vẫn có 1 số phần khác trong cài đặt

- Khai báo cấu trúc 1 process sẽ có thêm thuộc tính priority

```
//Khai bao cau truc sub
typedef struct{
    int head, tail;
}sub;

//Khai bao cau truc 1 process
typedef struct{
    int Name_Process; //1->n
    int CPU_Time;
    int Arrival_Time;
    int Priority;
    int State[LengthChart];
    int Waiting_Time;
    int Respose_Time;
    int Turnaround_Time;
    sub a[NB_Process];
    int sizePoint;
}Process2;
```

- Vẫn phải là mảng head, tail vì đây là Premtive
- Khai báo danh sách các mảng Process

```
//Khai bao cau truc danh sach tien trinh
typedef struct{
    Process2 process[NB_Process];
    int n;
}ListProcess2;
```

- Các hàm hỗ trợ cho danh sách process

- void initListProcess2(ListProcess2 *list): khởi tạo danh sách
- int isEmpty(ListProcess2 list): kiểm tra danh sách rỗng ?
- int isFull(ListProcess2 list, int nbProcess): kiểm tra danh sách đầy?
- void appendListProcess2(ListProcess2 *list, Process2 x): thêm một tiến trình vào ListProcess
- void deleteListProcess2(ListProcess2 *list, int index): xóa tiến trình tại index
- void Input2(ListProcess2 *list): nhập dữ liệu
- Process2 getNextProcess(ListProcess2 list): chọn tiến trình kế tiếp để thực hiện
- int PtoIndex(Process2 p, ListProcess2 list): trả về vị trí của tiến trình trong ListProcess
- int member1(Process2 x, ListProcess2 result): kiểm tra thành viên

- int smallest_Priority_current(ListProcess2 list): trả về tên của tiến trình có priority nhỏ I
  - Hàm PriorityPremtive

```
void Priority_Premtive(int Chart[], int *index, ListProcess2 *list, float *AVG_T, float *AVG_W, float *AVG_R, int *start){
    *AVG_T = 0;
    *AVG_W = 0;
    *AVG_R = 0;

    //Tim tien trinh den dau tien
    int i, j, min_Arrival_Time = INF, position;
    for(i = 0; i < list->n; i++){
        if(list->process[i].Arrival_Time <= min_Arrival_Time){
            min_Arrival_Time = list->process[i].Arrival_Time;
            position = i;
        }
    }

    //Them tien trinh vao LitsProcess va *index bat dau tu vi tri nay
    ListProcess2 result; initListProcess2(&result);
    appendListProcess2(&result, list->process[position]);
    *start = list->process[position].Arrival_Time;
    *index = list->process[position].Arrival_Time;

    //Khoi tao sub
    for(i = 0; i < list->n; i++){
        list->process[i].sizePoint = 0;
    }
```

- Các việc như khởi tạo, tìm tiến trình đầu tiên vào danh sách ListProcess, gán các giá trị *start

*index, vào khởi tạo mảng các cặp head, tail thì tương tự như thuật toán RR bên trên nhưng ta

không sử dụng hàng đợi ta sẽ sự dụng luôn cấu trúc ListProcess.

- Vòng lặp chính

```
    //Loop
    int k, have_process_come, count_seconds, donefill;
    while(!isEmpty(result)){
        Process2 p = getNextProcess(result);
        have_process_come = 0;
        //Tham chieu den ListProcess chinh
        int pos_root = PtoIndex(p, *list);
        //Tham chieu den ListProcess phu (chu yeu de xoa)
        int pos_child = PtoIndex(p, result);
        //Xac dinh sub(head)
        list->process[pos_root].a[list->process[pos_root].sizePoint].head = *index;
        donefill = 0;
```

- Lặp cho đến khi mảng các tiến trình result rỗng
- Tiến trình đầu tiên được lấy ra, với hàm getNextProcess thì tiến trình nào có priority nhỏ nhất sẽ được lấy ra
- Biến have_process_come kiểm tra xem có tiến trình nào tới hay không

- Biến pos_root có mục đích là tìm xem tiến trình đó nằm ở đâu trong ListProcess chính bởi vì ta cần chỉ số đó mới có thể cập nhật được thông số CPU Time ở root mà ở đây là ListProcess
- Biến pos_child thì chỉ chủ yếu để xóa và kiểm tra điều kiện vòng lặp chính
- Xác định head, tail cho tiến trình tại vị trí sizePonit mà được cấp phát ở trên
- Biến donefill là biến xác định xem tiến trình đó có được thực hiện xong hay chưa

- Trong vòng lặp con của vòng lặp chính

```
while(list->process[pos_root].CPU_Time+1 != 0){
    if(list->process[pos_root].CPU_Time == 0){
        if(donefill == 0){
            list->process[pos_root].a[list->process[pos_root].sizePoint].tail = *index;
        }
        //Danh dau xong tien trinh
        list->process[pos_root].CPU_Time = done;
        //Xoa tien trinh khoi danh sach
        deleteListProcess2(&result, pos_child);
        //Thoat
        break;
    }
    Chart[*index] = list->process[pos_root].Name_Process;
    list->process[pos_root].CPU_Time -= 1;
    *index += 1;
    //Duyet tim kiem tien trinh den trong thoi gian do (*index)
    for(i = 0; i < list->n; i++){
        if(list->process[i].Arrival_Time <= *index && !member1(list->process[i], result) && list->process[i].CPU_Time != don
            appendListProcess2(&result, list->process[i]);
            //Ton tai tien trinh toi
            have_process_come = 1;
        }
    }
        //Neu co tien trinh den thi ngat Tail, neu CPU_Time > 0 cap phat them sub
        if(have_process_come){
            if(smallest_Priority_current(result) != list->process[pos_root].Name_Process){
                list->process[pos_root].a[list->process[pos_root].sizePoint].tail = *index;
                donefill = 1;
                if(list->process[pos_root].CPU_Time > 0){
                    list->process[pos_root].sizePoint += 1;
                }
                break;
            }
        }
    }
```

- Lặp trong CPU Time+1 của tiến trình
- Nếu CPU Time của tiến trình đó chưa = 0 thì thêm vào mảng Chart[*index] = tên của tiến trình
- Cập nhật CPU Time của tiến trình
- Cập nhật *index
- Vòng lặp for là tìm kiếm tiến trình coi trong giây đó có tiến trình nào tới không nếu có thì thêm vào mảng result với điều kiện “CPU Time <= *index + Không phải thành viên của result và tiến trình đó thực hiện chưa xong khác done, và gán have_process_come = 1 báo hiệu là có tiến trình tới ta phải đến bước kế tiếp xét coi tiến trình nào có độ ưu tiên cao nhất

- Nếu mà các tiến trình vừa thêm ở bước trên khi gọi hàm smallest_Priority_curent(result) tức là tiến trình có priority nhỏ nhất hiện trong mảng tiến trình result không phải là tiến trình mình đang xét thì mình sẽ tính tail cho tiến trình đó và gán điểm chặn tail donefill = 1 và break khỏi vòng lặp con của vòng lặp chính và tìm xét tiếp tiến trình tiếp theo.
- Nếu mà CPU Time của tiến trình đã = 0 nếu điểm chặn donefill = 0 thì ta luôn tính luôn tail, đánh dấu tiến trình đó đã xong = done và xóa tiến trình đó ra khỏi ListProcess của result, quá trình cứ lặp đi lặp lại cho đến khi result rỗng

  - Tính các giá trị cần thiết

```
//Tinh cac du kien can thiet
for(i = 0; i < list->n; i++){
    *AVG_T += list->process[i].a[list->process[i].sizePoint].tail - list->process[i].Arrival_Time;
    *AVG_R += list->process[i].a[0].head - list->process[i].Arrival_Time;
    for(j = 0; j <= list->process[i].sizePoint; j++){
        if(j == 0){
            *AVG_W += list->process[i].a[0].head - list->process[i].Arrival_Time;
        }
        else{
            *AVG_W += list->process[i].a[j].head - list->process[i].a[j-1].tail;
        }
    }
}
*AVG_T /= list->n;
*AVG_R /= list->n;
*AVG_W /= list->n;
```

- Tương tự như là cách tính của thuật toán RoundRobin

  - Xử lý chuỗi định thời

```
//Xu ly du kien tren mang Chart[]
int z;
for(i = 0; i < list->n; i++){
    for(j = *start; j <= *index; j++){
        if( j >= list->process[i].Arrival_Time && j < list->process[i].a[list->process[i].sizePoint].tail){
            if(list->process[i].Name_Process == Chart[j]){
                list->process[i].State[j] = 'R';
            }
            else list->process[i].State[j] = 'W';
        }
        else
        list->process[i].State[j] = '-';
    }
}
```

  - Tương tự như là các tính của thuật toán RoundRobin

  - Hàm main

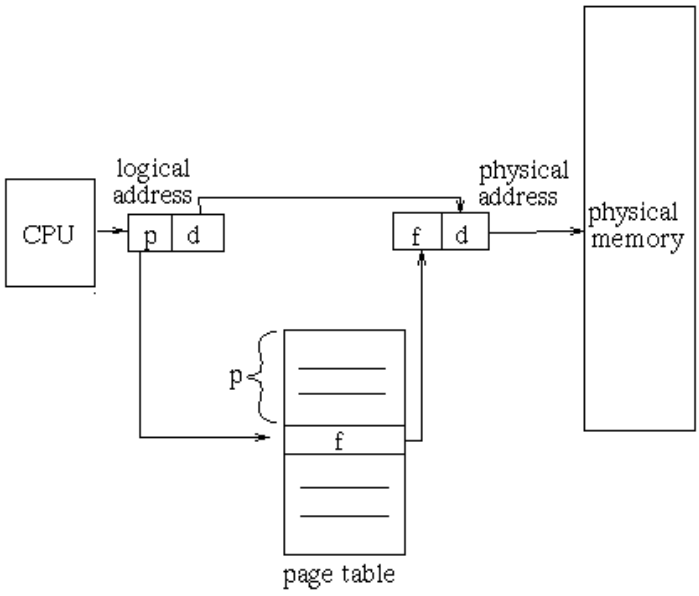
```
int chart[LengthChart], index, q, i, j, start;
float avgT, avgR, avgW;
ListProcess2 list;
Input2(&list);
Priority_Premtive(chart, &index, &list, &avgT, &avgW, &avgR, &start);
printf("Grantt Chart = [ ");
for(i = start; i <= index; i++){
    printf("%d ", chart[i]);
}
printf("]");
printf("\nDinh Thoi Priority Premtive\n");
for(i = 0; i < list.n; i++){
    for(j = start; j <= index; j++){
        printf("%c ", list.process[i].State[j]);
    }
    printf("\n");
}
printf("AVGW = %.2f\tAVGR = %.2f\tAVGT = %.2f",avgW, avgR, avgT);
printf("\n*****************************************\n");
```

- Chạy chương trình

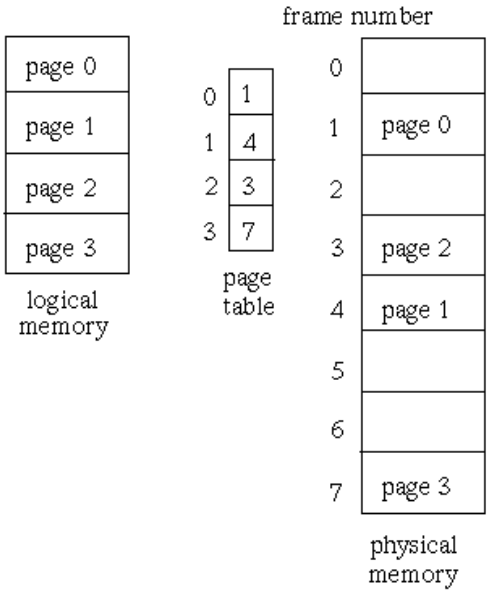
```
-----------------------------------MENU ALGORITHM-----------------------------------
                        1. FSFS algorithm
                        2. RounRobin algorithm
                        3. PriorityPremtive algorithm
                        0. exit


Choose the algorithm which you want: 3
Input number of process: 5
Input arrival time of the process 1 come system: 8
Input CPU time of the process 1: 10
Input priority of the process 1: 3
Input arrival time of the process 2 come system: 9
Input CPU time of the process 2: 1
Input priority of the process 2: 1
Input arrival time of the process 3 come system: 1
Input CPU time of the process 3: 3
Input priority of the process 3: 4
Input arrival time of the process 4 come system: 2
Input CPU time of the process 4: 1
Input priority of the process 4: 5
Input arrival time of the process 5 come system: 3
Input CPU time of the process 5: 8
Input priority of the process 5: 2
Grantt Chart = [ 3 3 5 5 5 5 5 5 2 5 5 1 1 1 1 1 1 1 1 1 1 3 4 ]
Dinh Thoi Priority Premtive
- - - - - - - W W W W R R R R R R R R R - - -
- - - - - - - - R - - - - - - - - - - - - - -
R R W W W W W W W W W W W W W W W W W W R - -
- W W W W W W W W W W W W W W W W W W W W R -
- - R R R R R R W R R - - - - - - - - - - - -
AVGW = 9.00     AVGR = 5.00     AVGT = 13.60
*****************************************
```

PHẦN 4: QUẢN LÝ BỘ NHỚ

- Với sơ đồ



- Xét ví dụ



- Ta thấy rằng với page 0 đang ở vị trí 0 trong Logical memory, xét đến tại chỉ số 0 trong page table ta thấy rằng có frame = 1 nên tại Physical memory ta sẽ thêm vào vùng nhớ tại chỉ số bằng frame tức là bằng 1 page 0 này, cứ tiếp tục như vậy đối với page1, page2, page3, ta thấy rằng việc đó dễ dàng để tạo một chương trình như sau:

- Đầu tiên ta sẽ khai báo cấu trúc 1 vùng memory

```c
//Khai bao cau truc 1 memory
typedef struct{
    char c;
    int number_page;
}element;
```

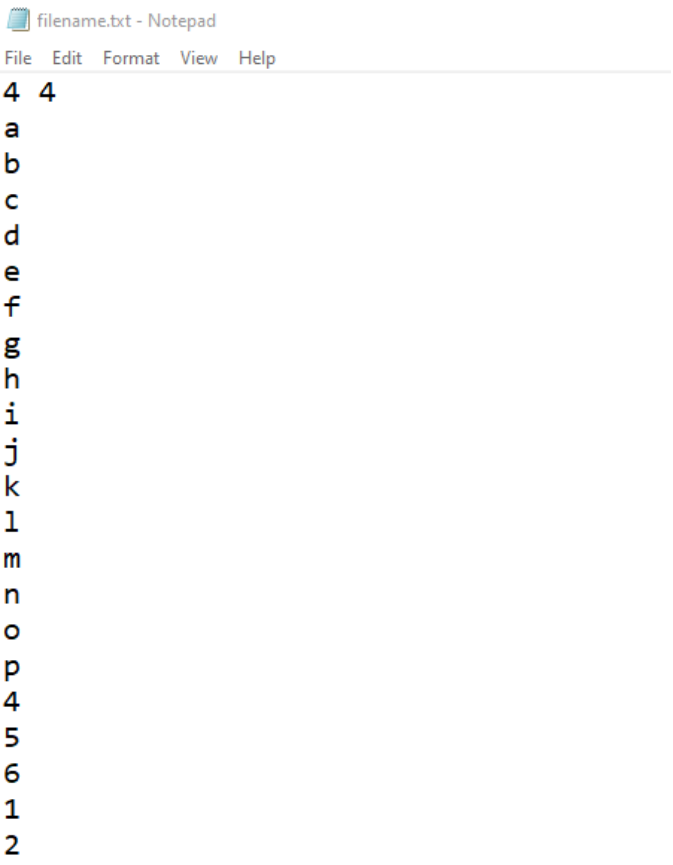
- Tiếp đến ta sẽ khai báo 1 mảng các vùng memory

```c
//Khai bao 1 mang memory
typedef struct{
    element M[MaxLength];
    int size;
}memory;

//Khoi tao mang memory
void initMemmory(memory *m){
    m->size = 0;
}

//Them phan tu
void addMemory(memory *m, char x){
    m->M[m->size].c = x;
    m->size++;
}
```

- Với hàm đọc dữ liệu đầu vào

```c
//Doc du lieu
void Input(memory *Logical, int PageTable[], int *indexPageTable, int *byte){
    initMemmory(Logical);
    freopen("filename.txt", "r", stdin);
    int numberPage, byteAPage;
    scanf("%d%d", &numberPage, &byteAPage);
    *byte = byteAPage;
    int i, count = 0, indexPage = 0;
    char c;
    for(i = 0; i < numberPage * byteAPage; i++){
        getchar();
        scanf("%c", &c);
        if(count == *byte){
            count = 0;
            indexPage++;
        }
        count++;
        addMemory(Logical, c);
        Logical->M[i].number_page = indexPage;
    }
    scanf("%d", &(*indexPageTable));
    int frame;
    for(i = 0; i < *indexPageTable; i++){
        scanf("%d", &frame);
        PageTable[i] = frame;
    }
}
```

- Ta cần filename.txt có định dạng như sau



- Hàm tìm maxval mục đích là để tạo size cho mảng Physical memory

```
//Tim max => length Physical
int max(int PageTable[], int indexPageTable){
    int maxval = -99999, i;
    for(i = 0; i < indexPageTable; i++){
        if(PageTable[i] > maxval){
            maxval = PageTable[i];
        }
    }
    return maxval;
}
```

- Hàm findIndexFrame mục đích là tìm kiếm index của Physical memory mà tại đó có frame tức là vùng sẽ được thêm vào từ Logical memory

```
//Tim index trong Physical
int findIndexFrame(int frame, memory Physical){
    int i;
    for(i = 0; i < Physical.size; i++){
        if(Physical.M[i].number_page == frame){
            return i;
        }
    }
}
```

- Hàm khởi tạo initPhysical khởi tạo các vùng frame và gán char tại đó bằng rỗng

```
//Khoi tao Physical value - frame
void initPhysical(int PageTable[], int indexPageTable, int byte, memory *Physical){
    int maxval = max(PageTable, indexPageTable);
    int length = maxval * byte + byte;
    Physical->size = length + byte;
    int i, count = 0, frame_nb = 0;
    for(i = 0; i < Physical->size; i++){
        if(count == byte){
            count = 0;
            frame_nb++;
        }
        Physical->M[i].c = ' ';
        Physical->M[i].number_page = frame_nb;
        count++;
    }
}
```

- Hàm LogicalToPhysical

```
void LogicalToPhysical(memory Logical, int PageTable[], int indexPageTable, memory *Physical, int byte){
    initPhysical(PageTable, indexPageTable, byte, Physical);
    int i;
    char CharAt;
    for(i = 0; i < Logical.size; i++){
        CharAt = Logical.M[i].c;
        int page = Logical.M[i].number_page;
        int frame = PageTable[page];
        int offset = 0;
        int j;
        for(j = page*byte; j < page*byte+byte; j++){
            if(Logical.M[j].c == CharAt){
                break;
            }
            else offset++;
        }
        int indexStart = findIndexFrame(frame, *Physical);
        Physical->M[indexStart+offset].c = CharAt;
    }

}
```

- Việc đầu tiên ta sẽ khởi tạo Physical memory
- Duyệt từng phần tử trong Logical memory
- CharAt/ lấy phần tự tại vị trí i
- page/ Xét xem tại phần tử này là trang mấy ở trong Logical memory
- Khi có page/ ta sẽ tìm kiếm nó sẽ lưu ở frame mấy của Physical memory
  frame = PageTable[page]
- Biến offset là biến độ dời
- Ta sẽ lặp từ đầu tới cuối của trang đó với số byte đề bài cho để tính offset
- biến indexStart sẽ thực hiện công việc là tìm index trong Physical memory
- Tại mảng Physical memory tại vin trí indexStart + offset sẽ là vị trí mà charAt được thêm vào

- Hàm main

```
int main(){
    memory Logical, Physical;
    int pageTable[MaxLength], indexPageTable, byte;
    Input(&Logical, pageTable, &indexPageTable, &byte);
    LogicalToPhysical(Logical, pageTable, indexPageTable, &Physical, byte);
    int i;
    printf("\n  PHYSICAL MEMORY\n");
    for(i = 0; i < Physical.size; i++){
        printf("%d\t%c\t%d\n", i, Physical.M[i].c, Physical.M[i].number_page);
    }
    return 0;
}
```

- Chạy chương trình

```
  PHYSICAL MEMORY
0               0
1               0
2               0
3               0
4       i       1
5       j       1
6       k       1
7       l       1
8       m       2
9       n       2
10      o       2
11      p       2
12              3
13              3
14              3
15              3
16              4
17              4
18              4
19              4
20      a       5
21      b       5
22      c       5
23      d       5
24      e       6
25      f       6
26      g       6
27      h       6
28              7
29              7
30              7
31              7
```