CSCI-3753: Operating Systems Fall 2019

Abigail Fernandes

Department of Computer Science

University of Colorado Boulder

Programming Assignment 2

LKM – Quick Recap

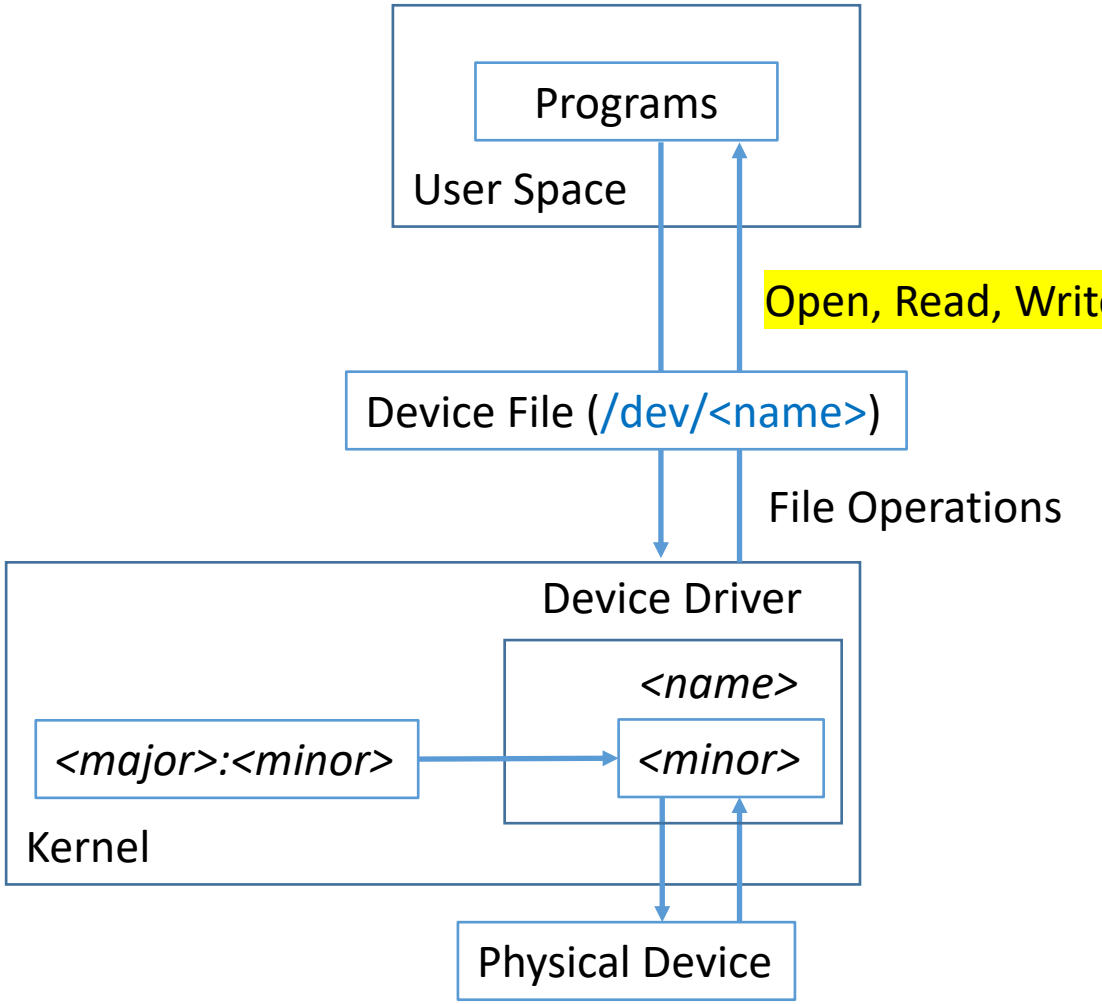
What are the two main functions required in a LKM code?

How do you compile the LKM code?

How do you load the module into the kernel?

How do you unload the module from the kernel?

PA2 – Character Device Driver



mknod –m <permission>
<device_file_location>
<type of driver>
<major number>
<minor number>

PA 2

1. Create a Device Driver Module (LKM)
2. Implement file operations (open, seek, read, write, release)
3. Make and load the module
4. Create a Device File for this Device
5. Test program that will allow you to perform different operations on the test file.

File Operations Data Structure

- Defined in /linux/fs.h

  vim /lib/modules/$(uname -r)/build/include/linux/fs.h

- An open device is identified internally by a file structure.
- The kernel uses the file_operations structure to access the driver's functions
- Each field in the structure must point to the function in the driver that implements a specific operation, or be left NULL for unsupported operations.

File Operations Data Structure

```
struct file_operations {
        struct module *owner;
        loff_t (*llseek) (struct file *, loff_t, int);
        ssize_t (*read) (struct file *, char __user *, size_t, loff_t *);
        ssize_t (*write) (struct file *, const char __user *, size_t, loff_t *);
        ssize_t (*read_iter) (struct kiocb *, struct iov_iter *);
        ssize_t (*write_iter) (struct kiocb *, struct iov_iter *);
        int (*iterate) (struct file *, struct dir_context *);
        int (*iterate_shared) (struct file *, struct dir_context *);
        unsigned int (*poll) (struct file *, struct poll_table_struct *);
        long (*unlocked_ioctl) (struct file *, unsigned int, unsigned long);
        long (*compat_ioctl) (struct file *, unsigned int, unsigned long);
        int (*mmap) (struct file *, struct vm_area_struct *);
        unsigned long mmap_supported_flags;
        int (*open) (struct inode *, struct file *);
        int (*flush) (struct file *, fl_owner_t id);
        int (*release) (struct inode *, struct file *);
        int (*fsync) (struct file *, loff_t, loff_t, int datasync);
        int (*fasync) (int, struct file *, int);
        int (*lock) (struct file *, int, struct file_lock *);
        ssize_t (*sendpage) (struct file *, struct page *, int, size_t, loff_t *, int);
        unsigned long (*get_unmapped_area)(struct file *, unsigned long, unsigned long, unsigned long, unsigned long);
        int (*check_flags)(int);
        int (*setfl)(struct file *, unsigned long);
        int (*flock) (struct file *, int, struct file_lock *);
        ssize_t (*splice_write)(struct pipe_inode_info *, struct file *, loff_t *, size_t, unsigned int);
        ssize_t (*splice_read)(struct file *, loff_t *, struct pipe_inode_info *, size_t, unsigned int);
        int (*setlease)(struct file *, long, struct file_lock **, void **);
        long (*fallocate)(struct file *file, int mode, loff_t offset,
                          loff_t len);
        void (*show_fdinfo)(struct seq_file *m, struct file *f);
#ifndef CONFIG_MMU
        unsigned (*mmap_capabilities)(struct file *);
#endif
        ssize_t (*copy_file_range)(struct file *, loff_t, struct file *,
                        loff_t, size_t, unsigned int);
        int (*clone_file_range)(struct file *, loff_t, struct file *, loff_t,
                        u64);
        ssize_t (*dedupe_file_range)(struct file *, u64, u64, struct file *,
                        u64);
} __randomize_layout;
```

System Device Major Numbers

vim /home/kernel/linux-hwe-4.15.0/include/uapi/linux/major.h

```
/* SPDX-License-Identifier: GPL-2.0 WITH Linux-syscall-note */
#ifndef _LINUX_MAJOR_H
#define _LINUX_MAJOR_H

/*
 * This file has definitions for major device numbers.
 * For the device number assignments, see Documentation/admin-guide/devices.rst.
 */

#define UNNAMED_MAJOR           0
#define MEM_MAJOR               1
#define RAMDISK_MAJOR           1
#define FLOPPY_MAJOR            2
#define PTY_MASTER_MAJOR        2
#define IDE0_MAJOR              3
#define HD_MAJOR                IDE0_MAJOR
#define PTY_SLAVE_MAJOR         3
#define TTY_MAJOR               4
#define TTYAUX_MAJOR            5
#define LP_MAJOR                6
#define VCS_MAJOR               7
#define LOOP_MAJOR              7
#define SCSI_DISK0_MAJOR        8
#define SCSI_TAPE_MAJOR         9
#define MD_MAJOR                9
#define MISC_MAJOR              10
#define SCSI_CDROM_MAJOR        11
#define MUX_MAJOR               11      /* PA-RISC only */
#define XT_DISK_MAJOR           13
#define INPUT_MAJOR             13
#define SOUND_MAJOR             14
#define CDU31A_CDROM_MAJOR      15
#define JOYSTICK_MAJOR          15
#define GOLDSTAR_CDROM_MAJOR    16
#define OPTICS_CDROM_MAJOR      17
#define SANYO_CDROM_MAJOR       18
#define CYCLADES_MAJOR          19
#define CYCLADESAUX_MAJOR       20
```

`/dev` Directory

```
$ ls -l /dev
brw-rw----   1 root disk      8,   0 Dec 20 20:13 sda
crw-rw-rw-   1 root root      1,   3 Dec 20 20:13 null
srw-rw-rw-   1 root root           0 Dec 20 20:13 log
prw-r--r--   1 root root           0 Dec 20 20:13 fdata
```

- Device Type
- Permissions
- Owner
- Group
- Major Device Number
- Minor Device Number
- Timestamp
- Device Name

Documentation for devices

See Documentation/admin-guide/devices.txt -> Current devices and major numbers

vim /home/kernel/linux-hwe-4.15.0/Documentation/admin-guide/devices.txt

```
1 char       Memory devices
               1 = /dev/mem          Physical memory access
               2 = /dev/kmem         Kernel virtual memory access
               3 = /dev/null         Null device
               4 = /dev/port         I/O port access
               5 = /dev/zero         Null byte source
               6 = /dev/core         OBSOLETE - replaced by /proc/kcore
               7 = /dev/full         Returns ENOSPC on write
               8 = /dev/random       Nondeterministic random number gen.
               9 = /dev/urandom      Faster, less secure random number gen.
              10 = /dev/aio          Asynchronous I/O notification interface
              11 = /dev/kmsg         Writes to this come out as printk's, reads
                                     export the buffered printk records.
              12 = /dev/oldmem       OBSOLETE - replaced by /proc/vmcore

1 block      RAM disk
               0 = /dev/ram0         First RAM disk
               1 = /dev/ram1         Second RAM disk
                 ...
             250 = /dev/initrd       Initial RAM disk
```

Creating Device File for a Device Driver

sudo mknod -m <permission> <device_file_location> <type_of_driver> <major_number> <minor_number>

Ex. sudo mknod -m 777 /dev/simple_character_device c 240 0

```
user@cu-cs-vm:/home/kernel/linux-hwe-4.15.0/Documentation/admin-guide$ sudo mknod -m 777 /dev/simple_character_device c 240 0
user@cu-cs-vm:/home/kernel/linux-hwe-4.15.0/Documentation/admin-guide$ ls /dev | grep simple
simple_character_device
user@cu-cs-vm:/home/kernel/linux-hwe-4.15.0/Documentation/admin-guide$ ls -l /dev | grep simple
crwxrwxrwx  1 root root    240,   0 Sep 19 22:59 simple_character_device
```

PA2 – Requirements

1. Dynamically allocate kernel buffer to store the data written by the user
   - kmalloc()
     - Allocate memory for objects smaller than page size in the kernel at initialization time
   - kfree()
     - Free memory previously allocated using kmalloc() before exiting

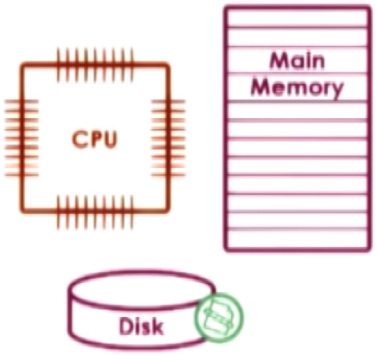
2. Offset pointer in read/write function

3. Use copy_to_user and copy_from_user inside the device driver.

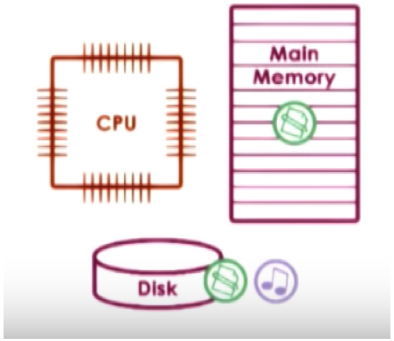
Processes and Threads

What is a process?

OS manages hardware on behalf of applications

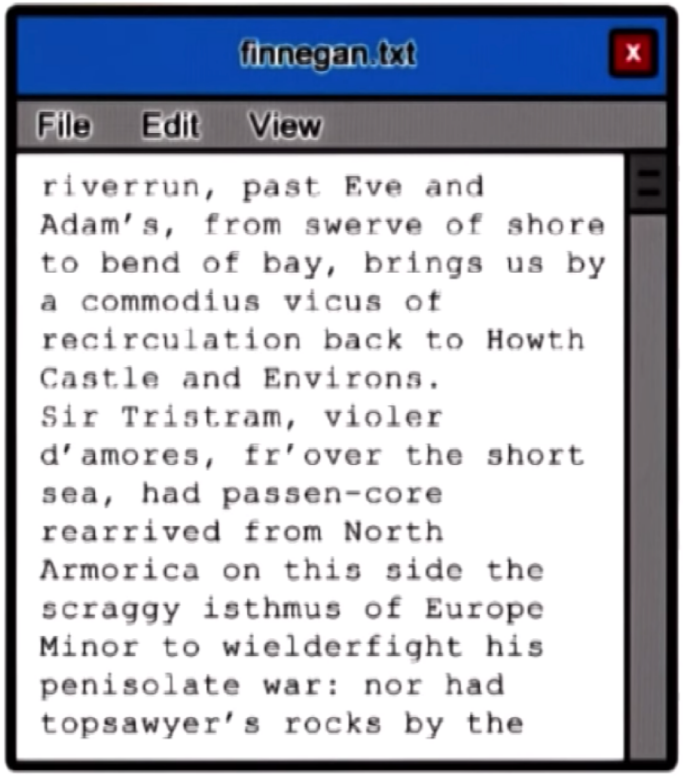


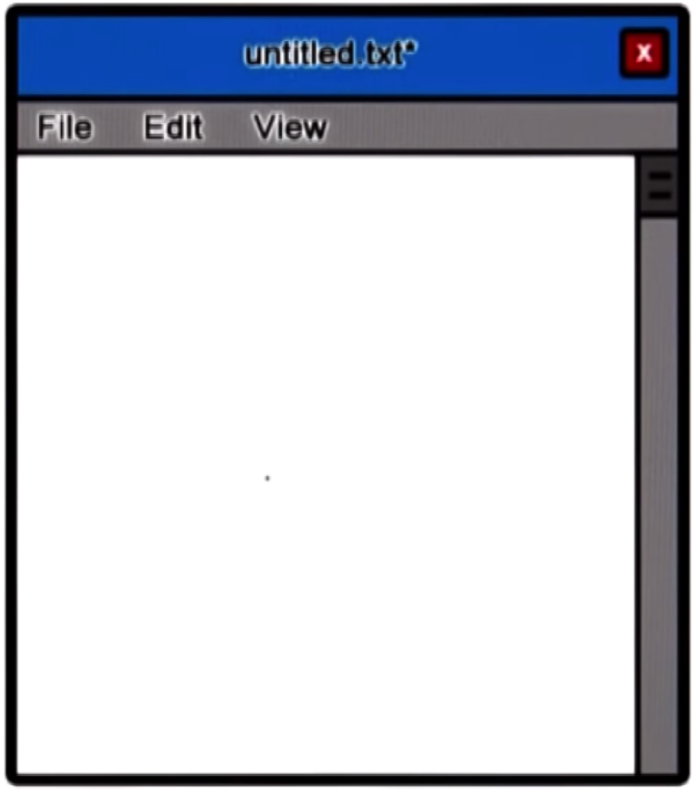
Application = program on disk, flash memory (static entity)



Process – state of a program when loaded in main memory (active entity)

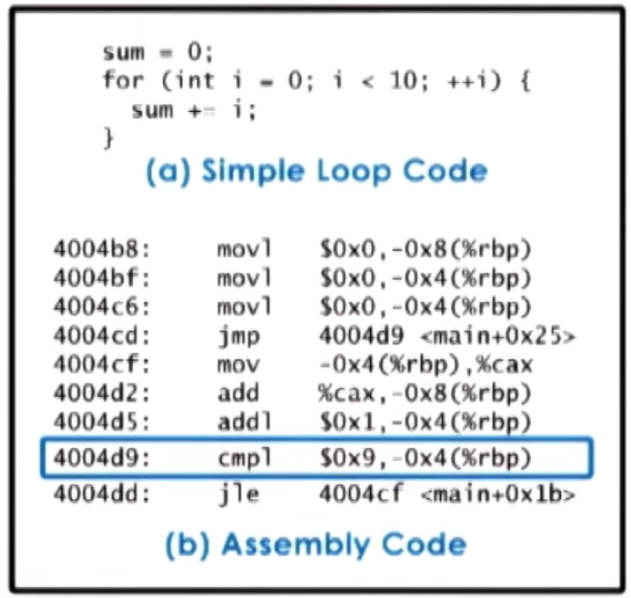
Example of a process

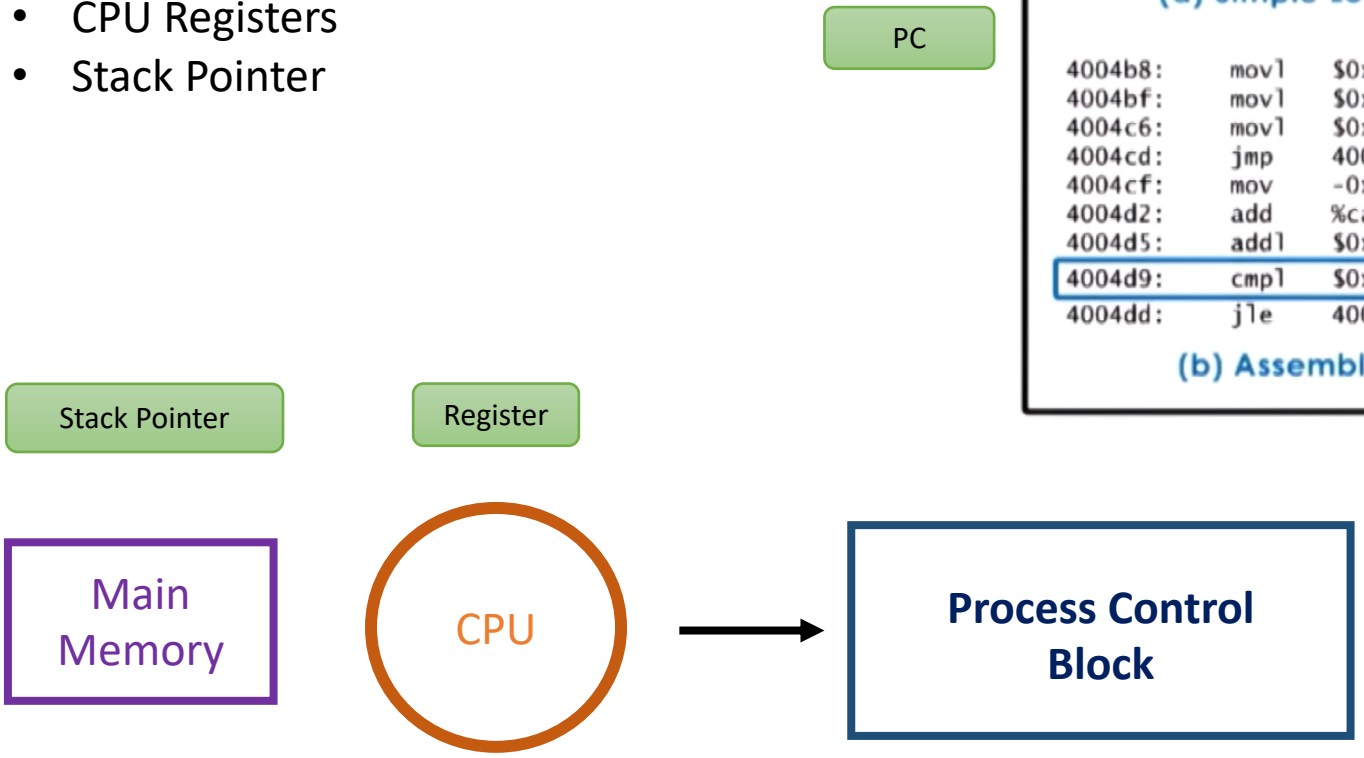




How does the OS know what a process is doing?

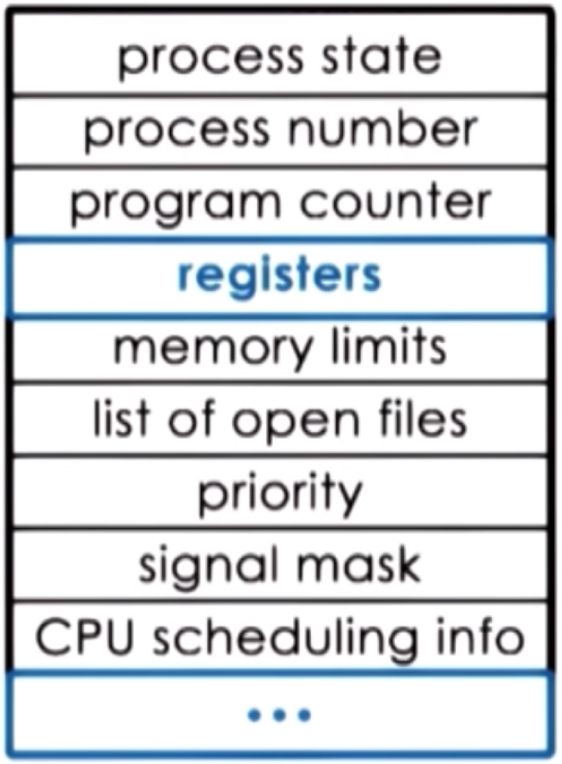
- Program Counter
- CPU Registers
- Stack Pointer



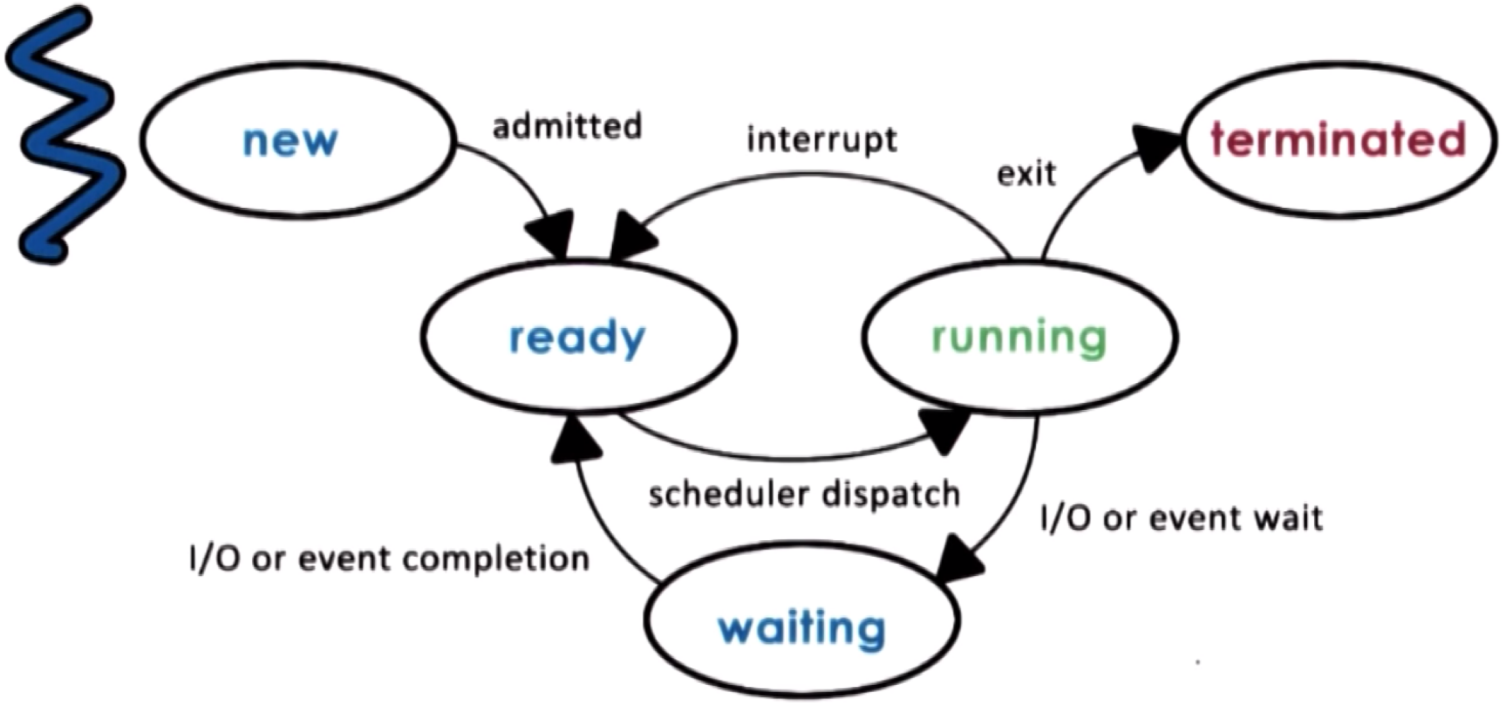


Process Control Block

- PCB created when process is created
- Certain fields are updated when process state changes
- Other fields change very frequently



Process Lifecycle: States



Process Lifecycle: States

New: The process is being created.

Running: Instructions are being executed.

Waiting: The process is waiting for some event to occur (such as an I/O completion or reception of a signal).

Ready: The process is waiting to be assigned to a processor.

Terminated: The process has finished execution.

Processes

Have an ID associated with it called the process ID (PID)

- **`ps`** command
  - PID: Process ID
  - TTY: Controlling terminal associated with the process
  - STAT: Process status code
  - TIME: Total CPU usage time
  - CMD: Name of executable/command

```
$ ps

PID        TTY      STAT   TIME           CMD
41230    pts/4    Ss       00:00:00      bash
51224    pts/4    R+       00:00:00      ps
```

/proc directory

```
user@cu-cs-vm:/home/kernel/linux-hwe-4.15.0/Documentation/admin-guide$ ls /proc
1     13    1407  1526   16718  177    19     202  219  25     30     42     843  957          fs           modules       thread-self
10    1334  1412  1540   17249  178    190    203  22   26     30805  43     844  96           interrupts   mounts        timer_list
1001  1339  1424  1545   17254  17853  1905   204  220  262    30836  6      85   971          iomem        mpt           tty
1019  1341  1427  1552   17255  17858  19059  205  221  263    31     7      852  acpi         ioports      mtrr          uptime
1024  1350  1443  1559   17258  179    191    206  222  264    32     70946  855  buddyinfo    irq          net           version
1025  1359  1449  1573   17259  18     192    207  223  26482  32694  72513  857  bus          kallsyms     pagetypeinfo  version_signature
1028  1370  1453  1575   17262  180    1924   208  224  265    327    72684  86   cgroups      kcore        partitions    vmallocinfo
105   1374  1457  1580   173    181    193    209  225  267    33     72917  87   cmdline      keys         sched_debug   vmstat
1050  1384  1458  1581   17337  182    194    21   226  268    34     72926  878  consoles     key-users    schedstat     zoneinfo
11    1388  1467  1584   17370  183    195    210  227  27     35     73003  88   cpuinfo      kmsg         scsi
1171  1390  1471  1593   174    184    196    211  228  27059  36     73096  880  crypto       kpagecgroup  self
12    1392  1476  16     17414  185    197    212  229  27061  364    73163  886  devices      kpagecount   slabinfo
122   1395  15    1664   17431  186    198    213  230  2743   369    73174  89   diskstats    kpageflags   softirqs
1222  1399  1504  16701  175    187    199    214  231  28     37     73187  9    dma          loadavg      stat
1278  14    1517  16710  176    188    2      215  232  28518  393    742    90   driver       locks        swaps
1284  1400  1520  16715  17682  18882  20     216  233  29     4      8      906  execdomains  mdstat       sys
1295  1402  1522  16716  17683  18886  200    217  234  290    405    829    907  fb           meminfo      sysrq-trigger
1297  1406  1523  16717  17686  189    201    218  24   291    41     838    929  filesystems  misc         sysvipc
```

```
File  Edit  View  Terminal  Tabs  Help
user@cu-cs-vm:/home/kernel/linux-hwe-4.15.0/Documentation/admin-guide$ sudo ls /proc/1
attr        cmdline          environ  io          mem         ns             pagemap      sched      smaps_rollup  syscall        wchan
autogroup   comm             exe      limits      mountinfo   numa_maps      patch_state  schedstat  stack         task
auxv        coredump_filter  fd       loginuid    mounts      oom_adj        personality  sessionid  stat          timers
cgroup      cpuset           fdinfo   map_files   mountstats  oom_score      projid_map   setgroups  statm         timerslack_ns
clear_refs  cwd              gid_map  maps        net         oom_score_adj  root         smaps      status        uid_map
```

/proc directory

- Contain virtual files and numbered directories corresponding to each running process
- Directory names = process ids
- When a process ends, its directory in /proc disappears automatically.
- Files in a numbered directory
  - cmdline
  - cwd
  - environ
  - fd
  - maps, statm, mem; stat, status

/proc directory

Some typical virtual files that provide

Hardware information

- /proc/cpuinfo: identifies the type of processor used by your system
- /proc/iomem: shows you the current map of the system's memory for each physical device
- /proc/meminfo
- /proc/interrupts

File-related info

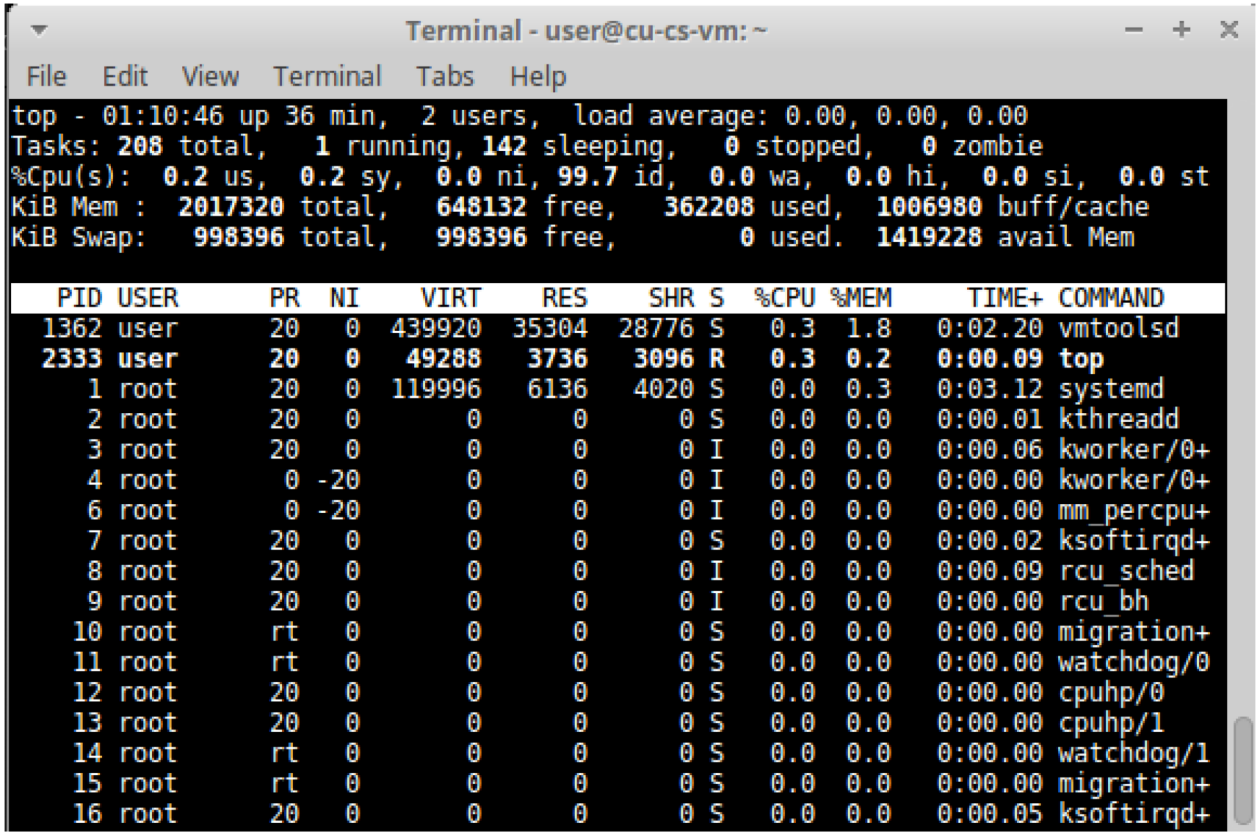
- /proc/filesystems: displays a list of the file system types currently supported by the kernel
- /proc/partitions

Kernel configuration parameters

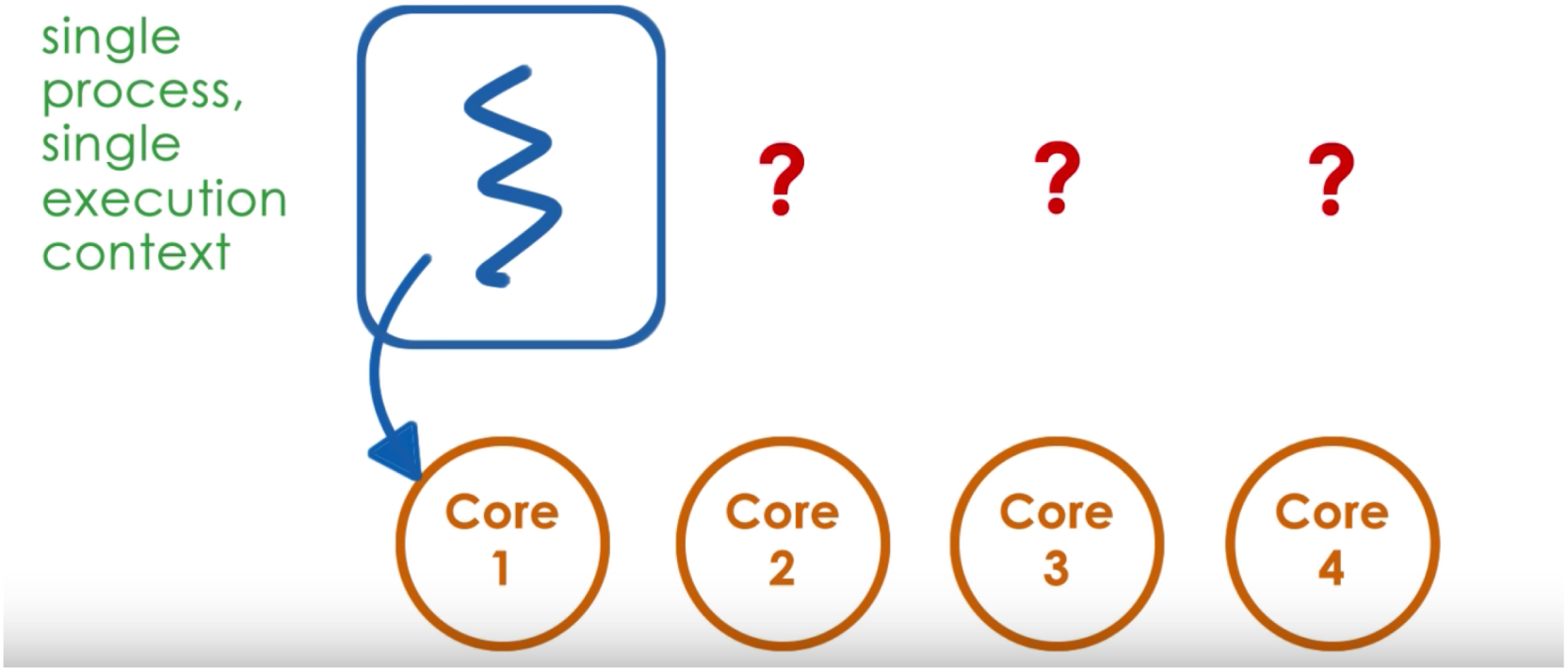
- /proc/cmdline: shows the parameters passed to the kernel at the time it is started
- /proc/sys

`top` command

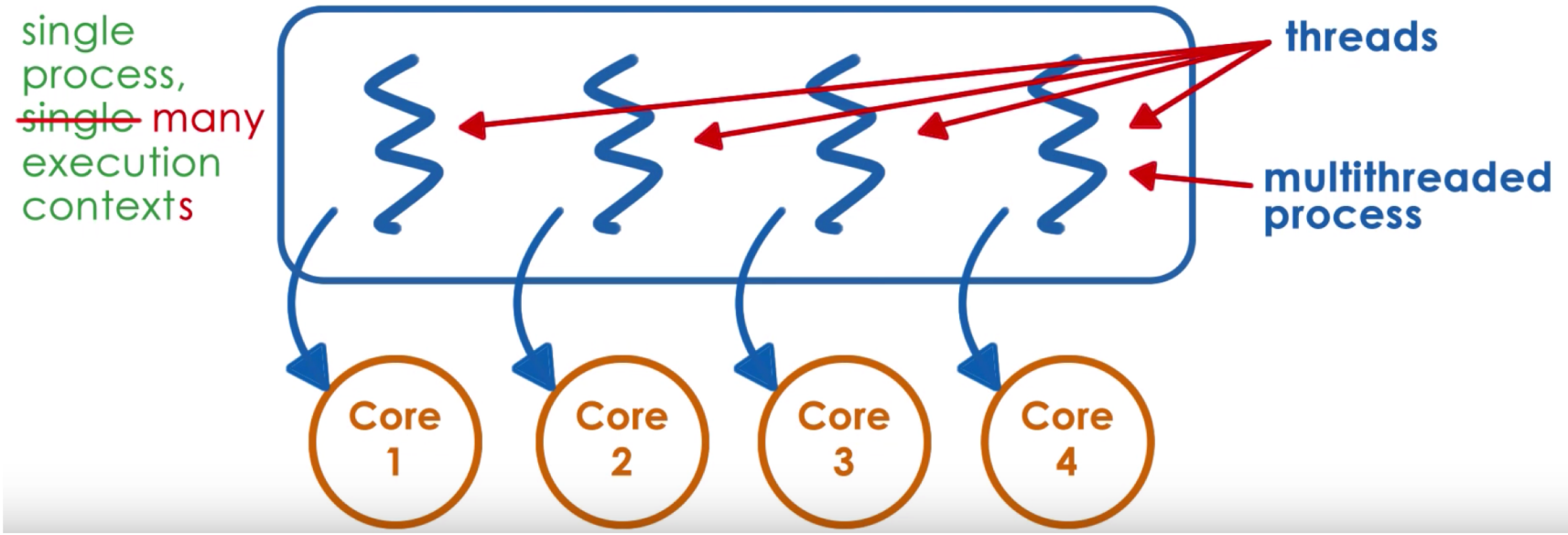
- Display a list of processes with their resource usage.



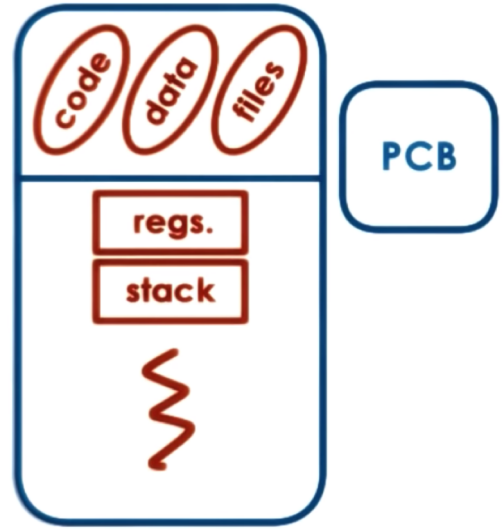
What if we have multiple CPUs

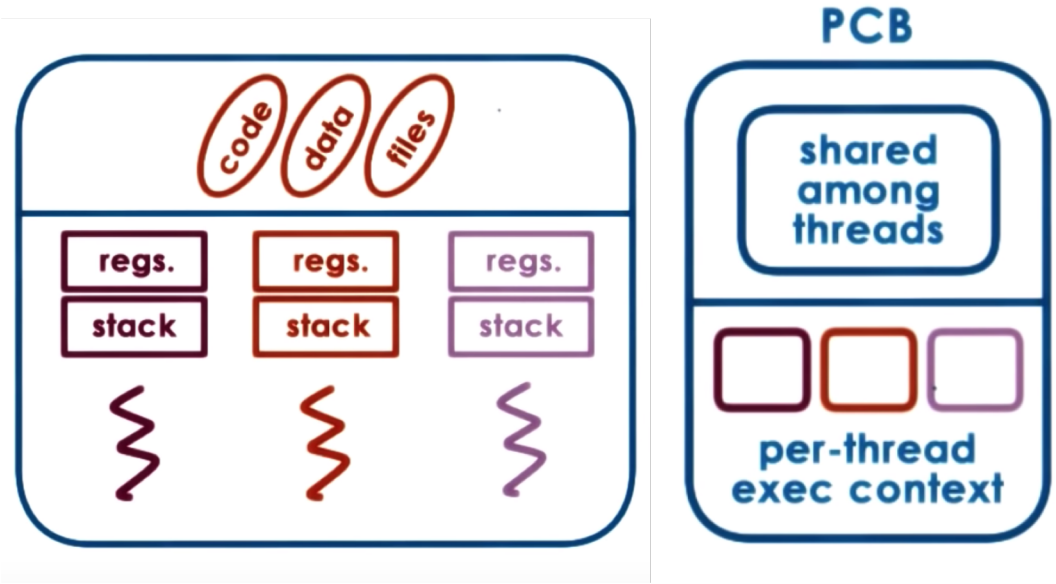


What if we have multiple CPUs



Process vs Threads





Process vs Thread

Process	Thread
Executing instance of an application	Path of execution within a process
Used for heavy weight tasks	Used for small tasks
Takes more time for termination and creation and context switching	Takes less time for termination and creation and context switching
Process consume more resources.	Threads consume less resources
Process is isolated	Threads share memory

Week 4 – Checklist

- ❑ Discuss PA2
- ❑ Discuss processes vs threads
- ❑ Read about IPC