University of Derby - BA, BSc, FdA, FdSc, MA, MSc, PhD Degrees & Short Courses - University of Derby

Operating Systems

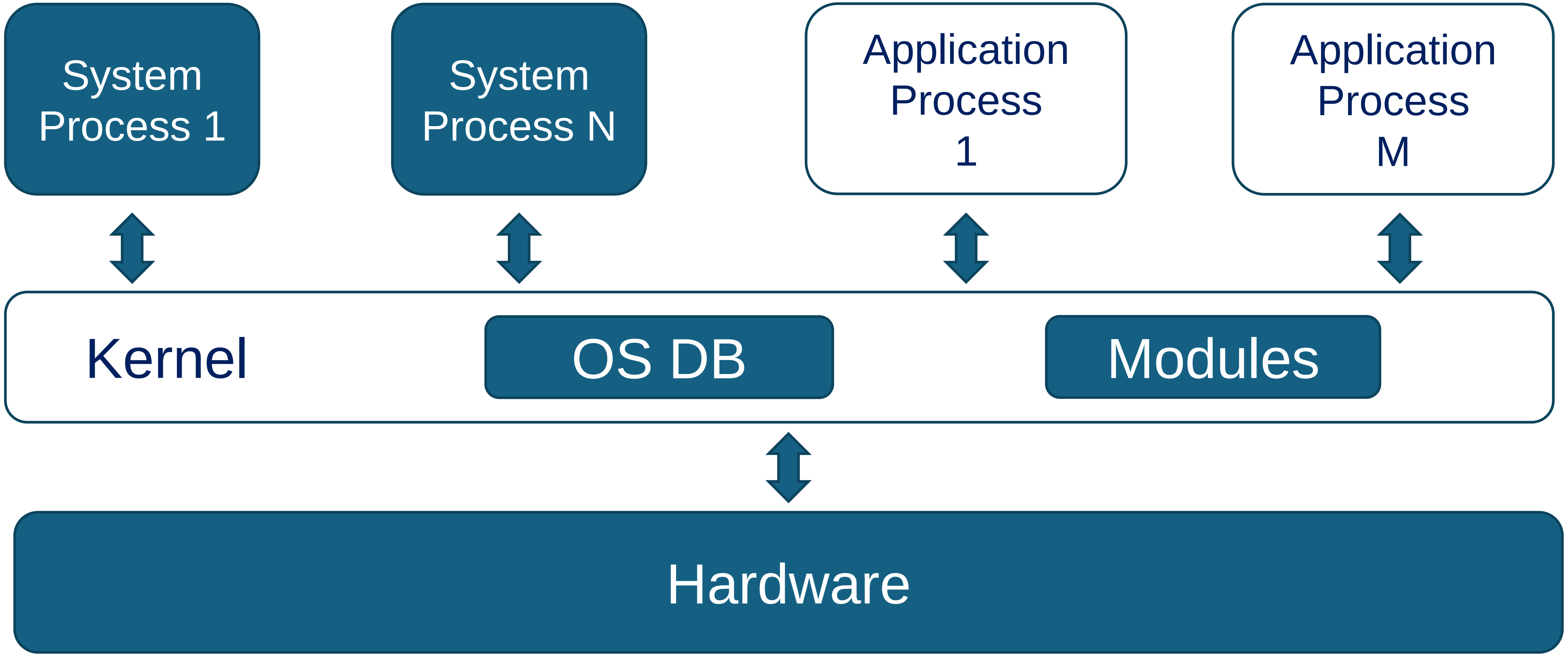
Dmitry Zaitsev

Lecture 3: Structure of OS: kernel & processes. Case study: Linux CLI, protection of files.

Basic components of OS

- **Kernel** – a part of OS which is not a process and, traditionally, resides in RAM, providing basic services, usually process scheduling, inter-process communication, memory control and I/O device control, represented with a series of device drivers
- **System processes** – components, which implement time-consuming service, for instance file operations, network communication, monitoring, planning, etc

Structure of OS



Kernel of OS

- Kernel is entered via interrupts: hardware interrupts – for device drivers and software interrupts – for system call processing
- Kernel consists of modules, which implement operations over resources, and a data base of OS
- Data base of OS represents a multi-linked list of control blocks: process control block, memory page table, device control block, controller control block, etc
- Usually, a uniform entry point is provided for system call implementation – dispatcher of system calls

Types of OS kernel

- Monolithic – a large, single kernel with all core operating system services running in a single, shared address space; high performance; low reliability
- Microkernel – a small, deliberately minimal kernel that provides only essential services, with other functions running in user space; higher reliability; reduced performance
- Hybrid – a combination of monolithic and microkernel; balanced performance and reliability
- *Nanokernel for minimal hardware management*
- *Exokernel exposes hardware directly*
- *Multikernel for distributed systems*

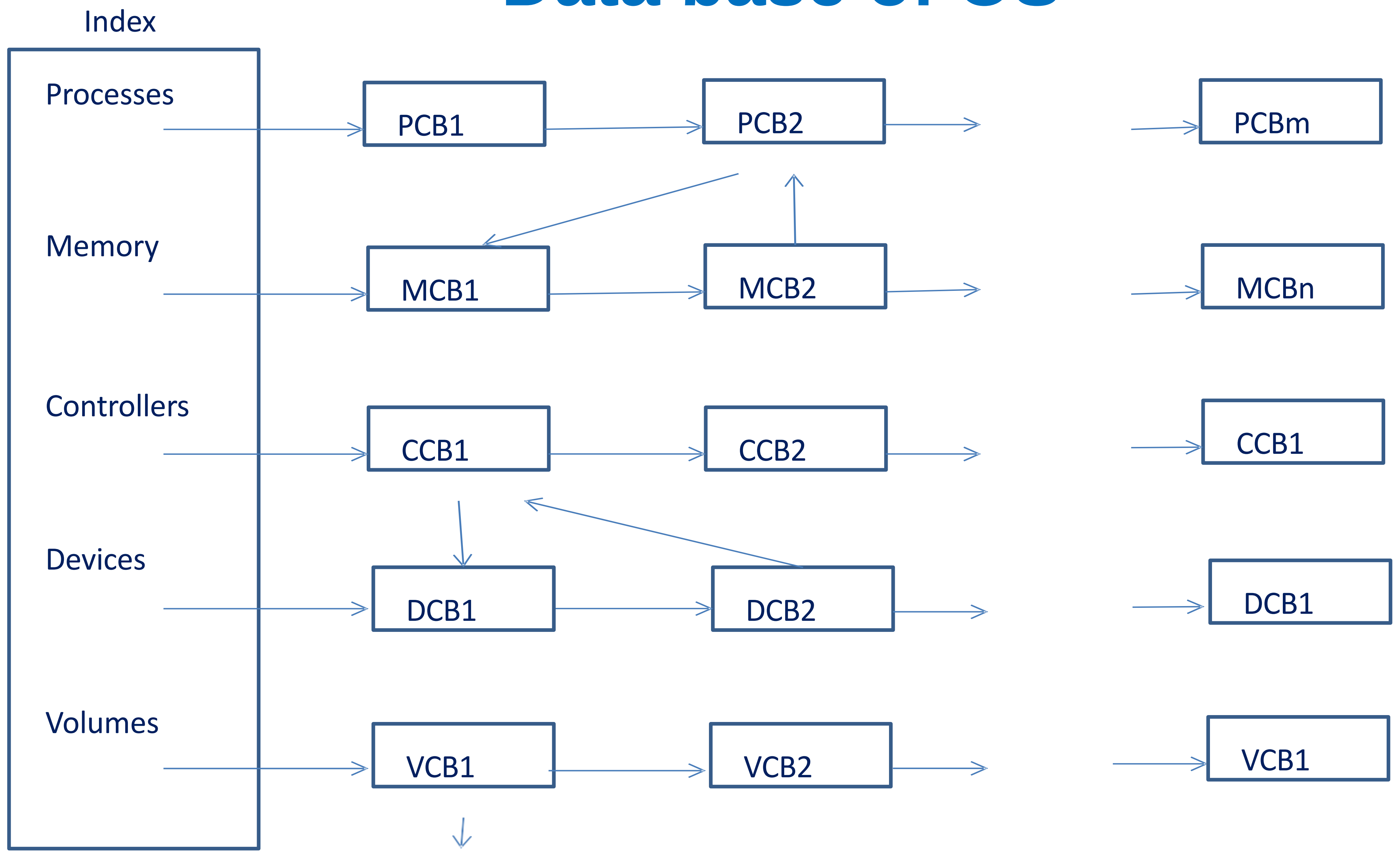
A concept of a process within OS

- Process is a computer application (program) being executed
- We can start a few processes to execute the same application file
- Basic stages of a process's life:
  - ✓Issue a CLI command or press GUI button
  - ✓Create PCB within OS DB
  - ✓Allocate initial resources
  - ✓Submit to the Process scheduler
  - ✓Run on CPU and process a sequence of system calls
  - ✓Deallocate resources
  - ✓Dispose of PCB

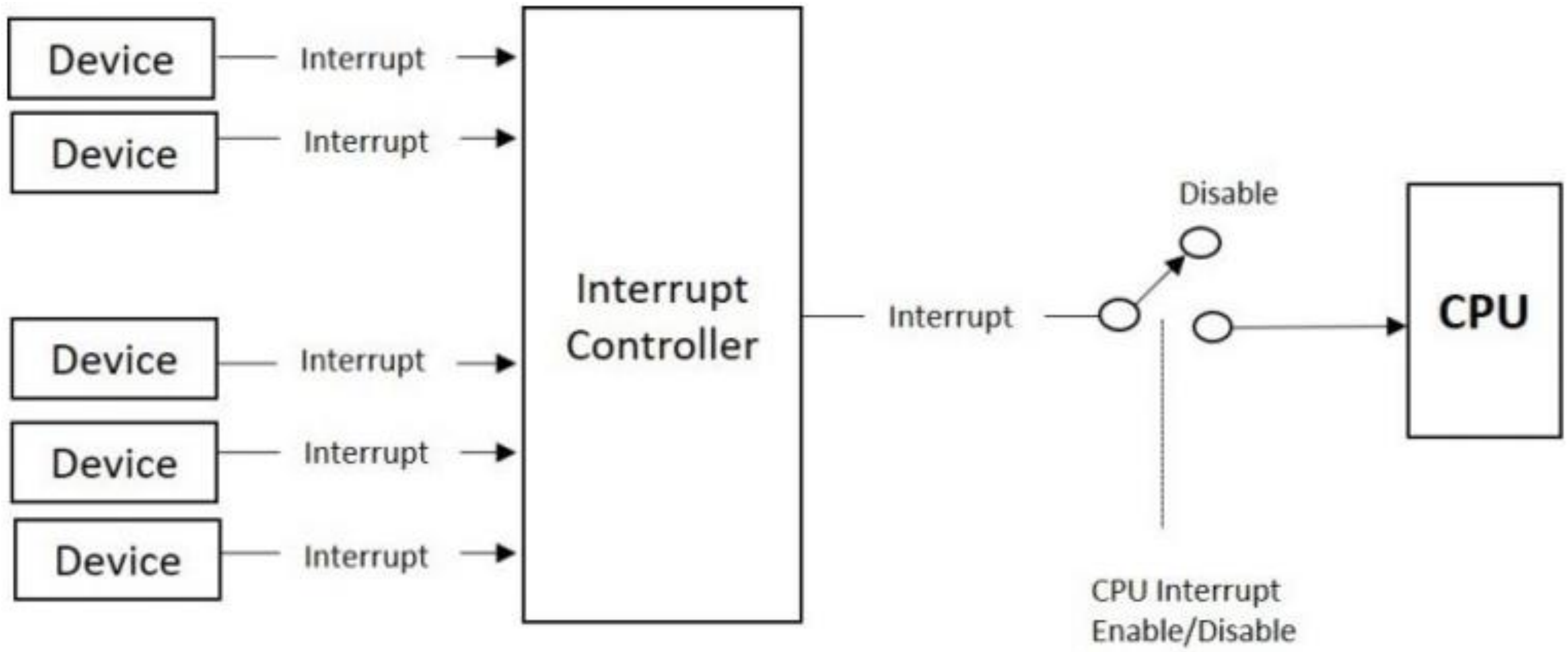
System process

- Represents a component of OS
- Rather independent from kernel timed, security, and other restrictions and context
- Can work at higher hardware privilege levels (than User process)
- Has access to OS DB
- Works on orders provided by OS kernel
- Processing specific events and requests, for example HTTP requests for a web-server
- Representing an extension for non time-critical system call implementation, for instance file operations

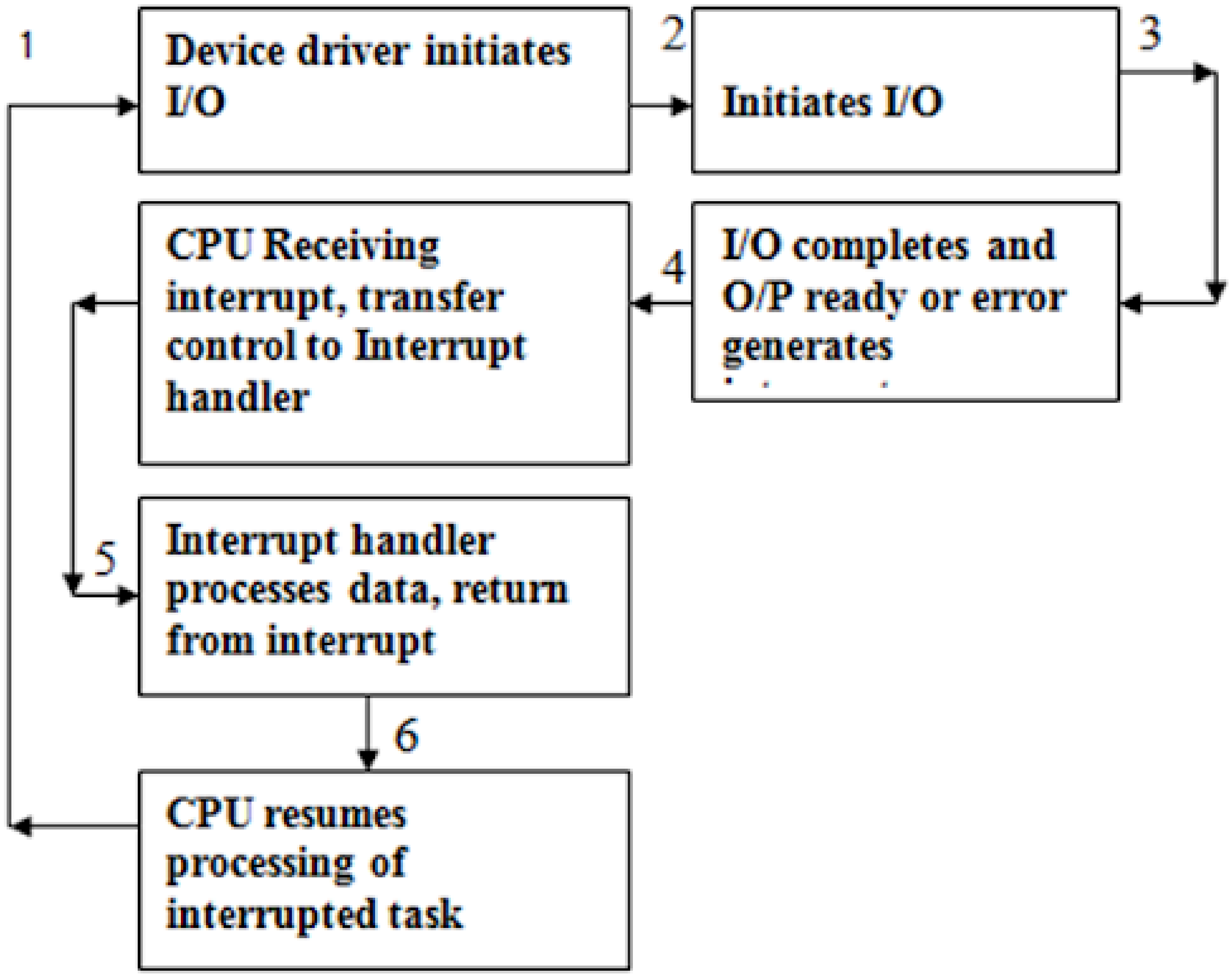
Data base of OS



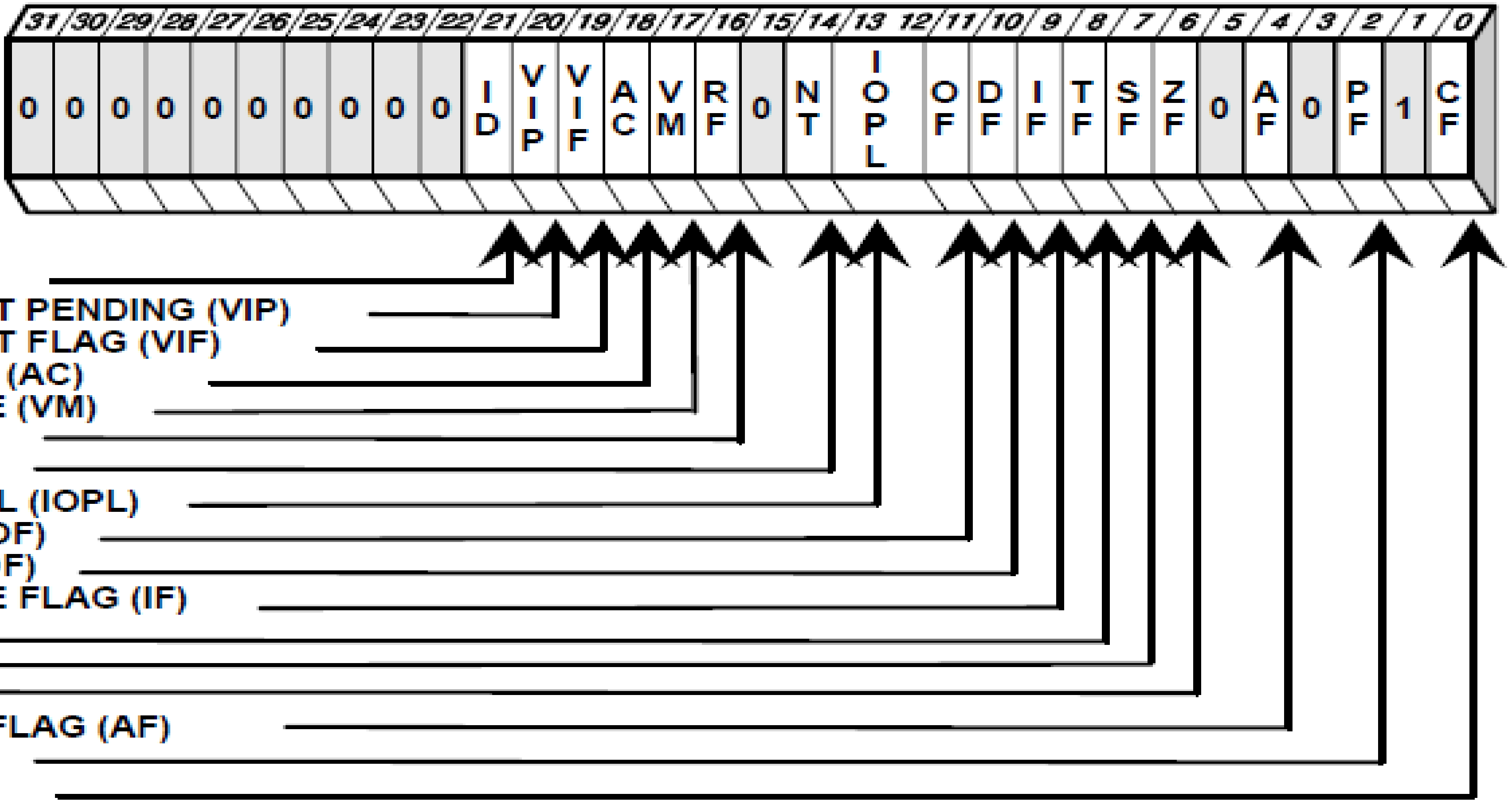
Event driven control with interrupts



Asynchronous I/O implementation via interrupts


1
Device driver initiates I/O
2
Initiates I/O
3
I/O completes and O/P ready or error generates
4
CPU Receiving interrupt, transfer control to Interrupt handler
5
Interrupt handler processes data, return from interrupt
6
CPU resumes processing of interrupted task

PSW of Pentium



X ID FLAG (ID)
X VIRTUAL INTERRUPT PENDING (VIP)
X VIRTUAL INTERRUPT FLAG (VIF)
X ALIGNMENT CHECK (AC)
X VIRTUAL 8086 MODE (VM)
X RESUME FLAG (RF)
X NESTED TASK (NT)
X I/O PRIVILEGE LEVEL (IOPL)
S OVERFLOW FLAG (OF)
C DIRECTION FLAG (DF)
X INTERRUPT ENABLE FLAG (IF)
X TRAP FLAG (TF)
S SIGN FLAG (SF)
S ZERO FLAG (ZF)
S AUXILIARY CARRY FLAG (AF)
S PARITY FLAG (PF)
S CARRY FLAG (CF)

S INDICATES A STATUS FLAG
C INDICATES A CONTROL FLAG
X INDICATES A SYSTEM FLAG

Security of OS

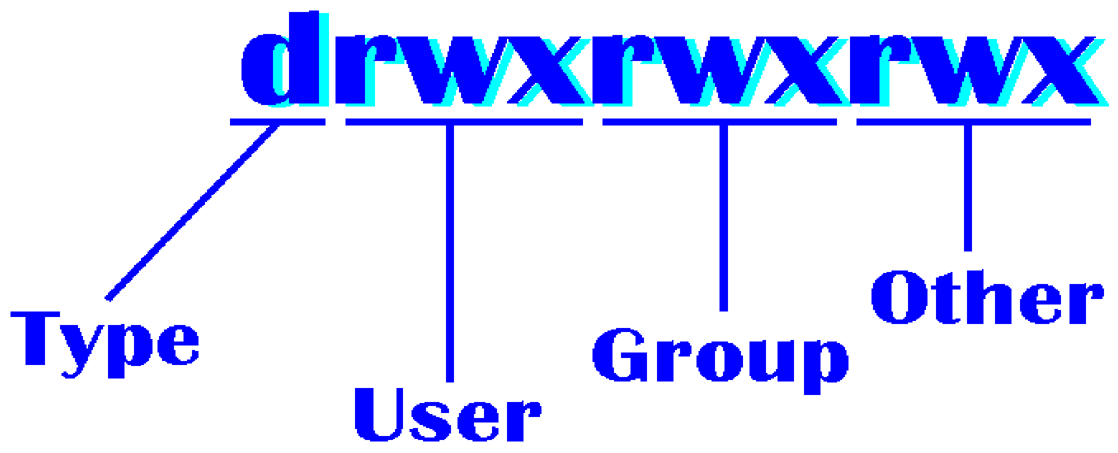
- Protect OS
- Protect Processes
- Protect Resources

- Hardware security support – modes of processor work, memory protection
- Identification of users
- Access rights (permissions)

Information protection in Linux

- Protection of volumes, directories, and files
- Each object has: user owner, group owner, permissions
- Permissions – 3 parts: User, Group, Other
- For each part of permissions – access rights: Read, Write, eXecute
- *user + s (pecial) or SUID: always executes as the user who owns the file*
- *group + s (pecial) or SGID: allows the file to be executed as the group that owns the file*
- *other + t (sticky): historical, for RAM resident processes*

Permissions scheme



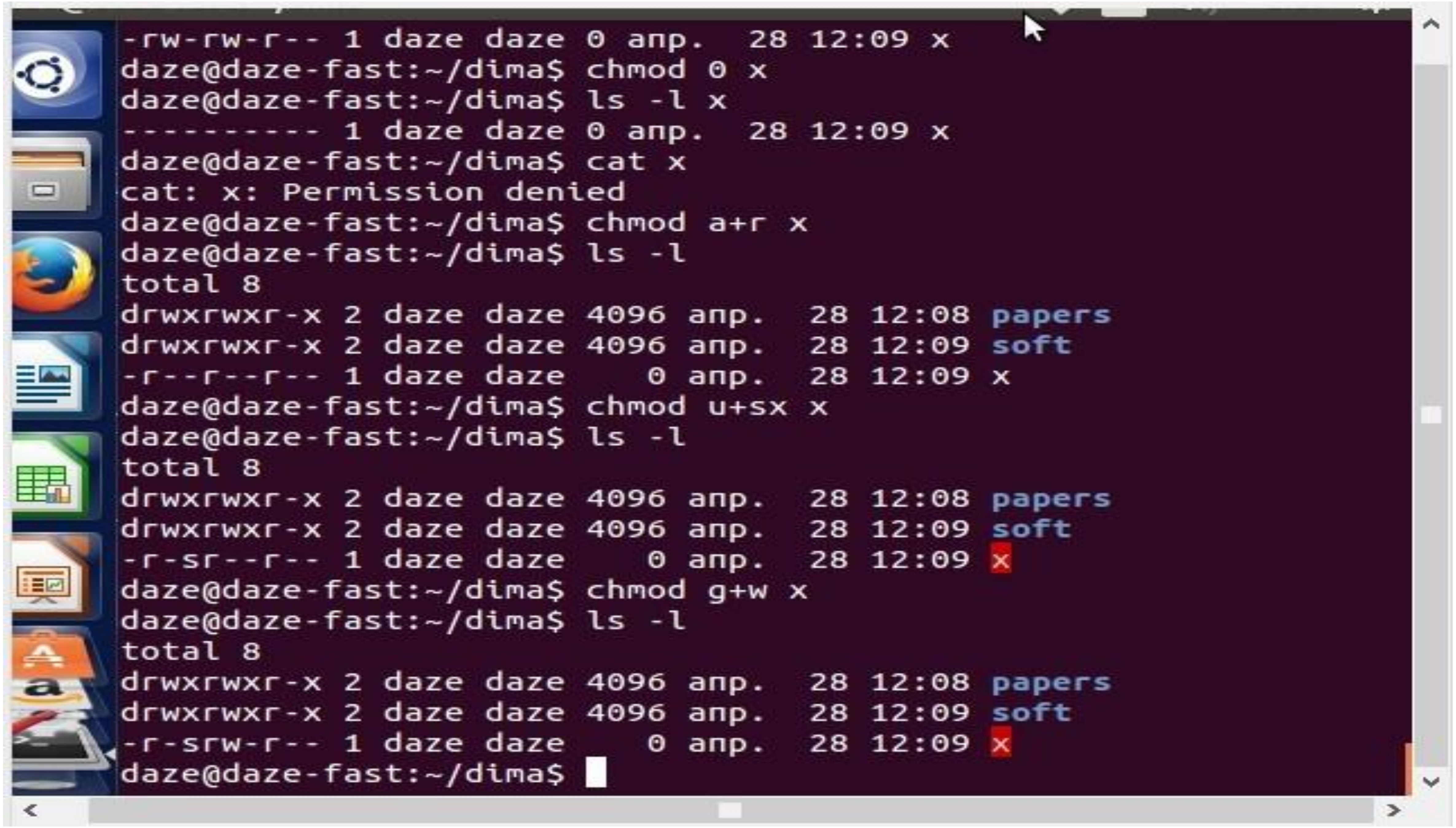
Octal	Decimal	Permission	Representation
000	0 (0+0+0)	No Permission	---
001	1 (0+0+1)	Execute	--x
010	2 (0+2+0)	Write	-w-
011	3 (0+2+1)	Write + Execute	-wx
100	4 (4+0+0)	Read	r--
101	5 (4+0+1)	Read + Execute	r-x
110	6 (4+2+0)	Read + Write	rw-
111	7 (4+2+1)	Read + Write + Execute	rwx

Change protection mode

- chmod [ugoa] [+-=] [rwxXst] file
- chmod octal-code file

- chmod u+w file1
- chmod a+r file2
- chmod 444 file3

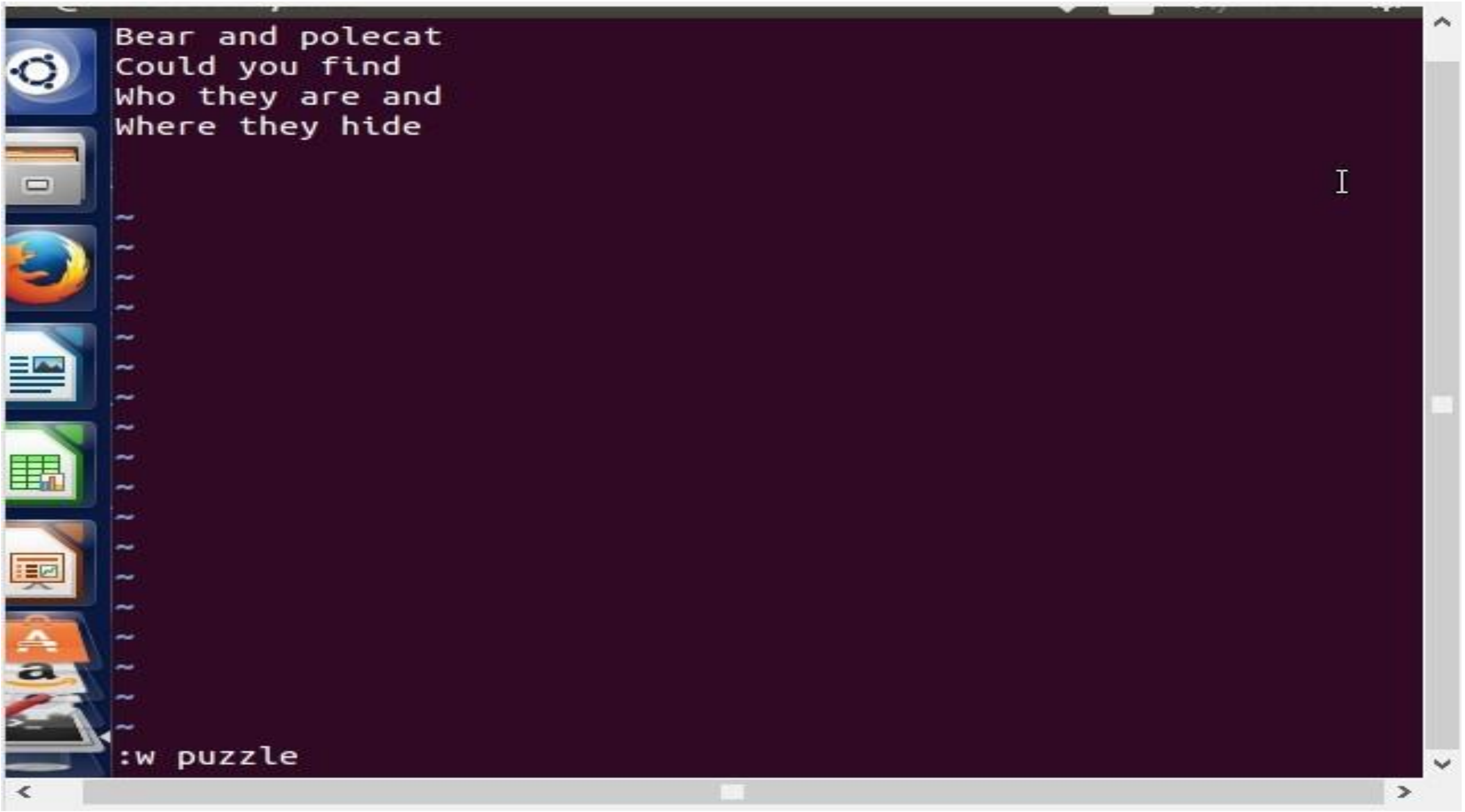
- For special bits – additional first digit: 4 – SUID, 2 – SGID, 1 - Sticky



```
-rw-rw-r-- 1 daze daze 0 апр.  28 12:09 x
daze@daze-fast:~/dima$ chmod 0 x
daze@daze-fast:~/dima$ ls -l x
---------- 1 daze daze 0 апр.  28 12:09 x
daze@daze-fast:~/dima$ cat x
cat: x: Permission denied
daze@daze-fast:~/dima$ chmod a+r x
daze@daze-fast:~/dima$ ls -l
total 8
drwxrwxr-x 2 daze daze 4096 апр.  28 12:08 papers
drwxrwxr-x 2 daze daze 4096 апр.  28 12:09 soft
-r--r--r-- 1 daze daze    0 апр.  28 12:09 x
daze@daze-fast:~/dima$ chmod u+sx x
daze@daze-fast:~/dima$ ls -l
total 8
drwxrwxr-x 2 daze daze 4096 апр.  28 12:08 papers
drwxrwxr-x 2 daze daze 4096 апр.  28 12:09 soft
-r-sr--r-- 1 daze daze    0 апр.  28 12:09 x
daze@daze-fast:~/dima$ chmod g+w x
daze@daze-fast:~/dima$ ls -l
total 8
drwxrwxr-x 2 daze daze 4096 апр.  28 12:08 papers
drwxrwxr-x 2 daze daze 4096 апр.  28 12:09 soft
-r-srw-r-- 1 daze daze    0 апр.  28 12:09 x
daze@daze-fast:~/dima$ 
```

How to create/edit textual file

- touch file
- nano file
- vi file
- vi modes: editing, editing command, file command
- i — Switch to Insert mode, Esc — Switch to Command mode
- More commands: a — Append after the cursor’s current position; x — Delete a single character; dd — Delete an entire line.
- File command: w file; r file; q; q!



```
Bear and polecat
Could you find
Who they are and
Where they hide



~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
:w puzzle
```

Work with volumes

- *mount* command displays all currently attached file systems (volumes)
- mount a volume
- *mount [OPTION...] DEVICE_NAME DIRECTORY*
- an example of mounting a volume
- *sudo mount /dev/sdb1 /mnt/media*
- list of typical volumes */etc/fstab*
- [File System] [Mount Point] [File System Type] [Options] [Dump] [Pass]
- unmount a volume
- *umount [OPTION...] DEVICE_NAME or DIRECTORY*