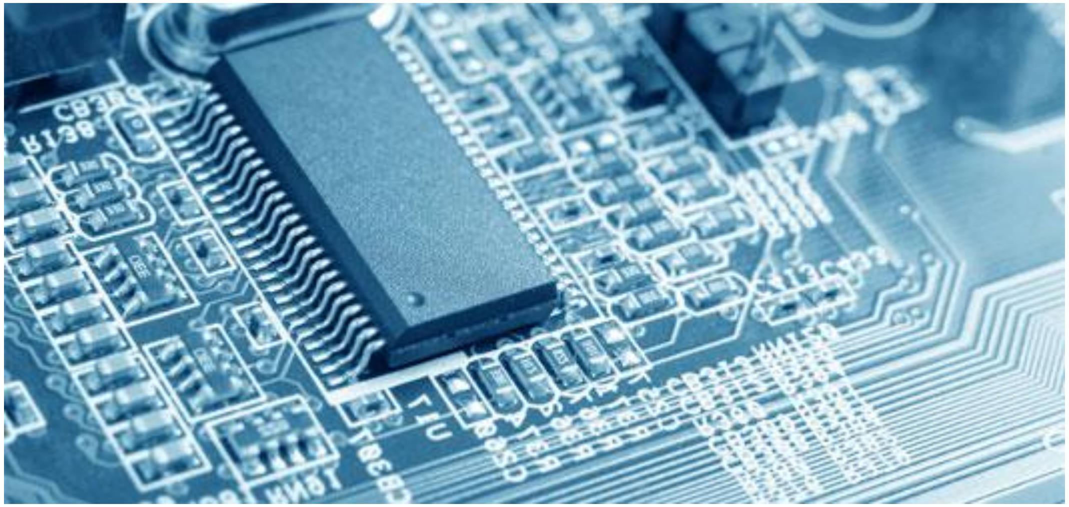
NAZARBAYEV UNIVERSITY
SCHOOL OF SCIENCE AND TECHNOLOGY



# ROBT305 - Embedded Systems

## Lectures 2-3 – Operating System Structure, Processes, Concurrency, Multithreaded Programming, Introduction to Linux OS

**20-25 August, 2015**

# Course Logistics

**Reading Assignment:  Chapters 2-4 of the** of the Operating Systems Concept textbook – slides relevant material only

# Operating Systems

- **An operating system** provides an environment for execution of programs and services to programs and users
- One set of operating system services provides functions that are helpful to the user:
  - **User interface -** Almost all operating systems have a user interface (UI).
    - Varies between **Command-Line (CLI)**, **Graphics User Interface (GUI)**, **Batch**
  - **Program execution -** The system must be able to load a program into memory and to run that program, end execution, either normally or abnormally (indicating error)
  - **I/O operations -** A running program may require I/O, which may involve a file or an I/O device
  - **File-system manipulation -** Programs need to read and write files and directories, create and delete them, search them, list file Information, permission management.

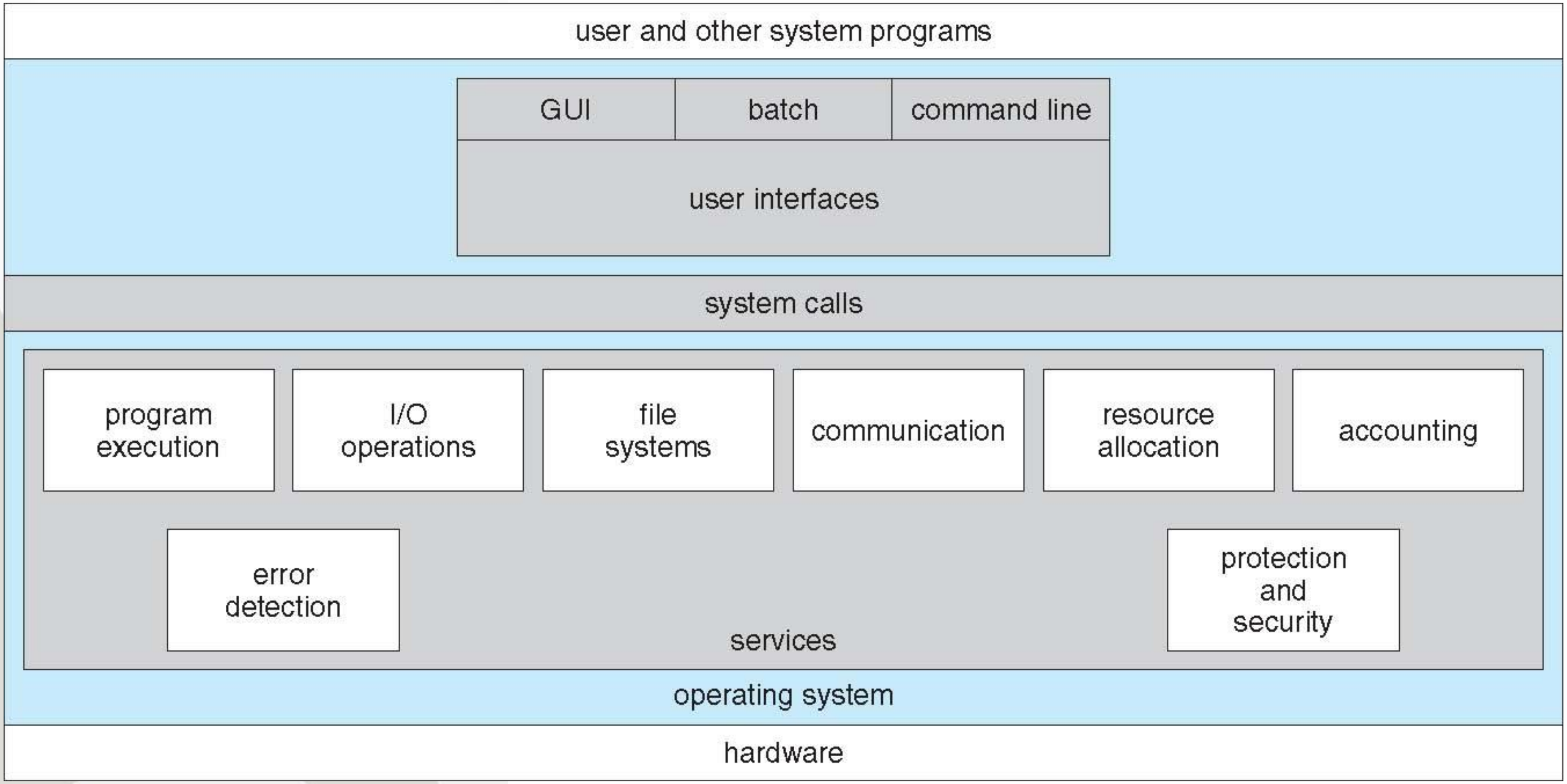
# Operating System (Cont.)

- **Communications** – Processes may exchange information, on the same computer or between computers over a network
  - Communications may be via shared memory or through message passing (packets moved by the OS)
- **Error detection** – OS needs to be constantly aware of possible errors
  - May occur in the CPU and memory hardware, in I/O devices, in user program
  - For each type of error, OS should take the appropriate action to ensure correct and consistent computing
  - Debugging facilities can greatly enhance the user's and programmer's abilities to efficiently use the system

# Operating System Services (Cont.)

- Another set of OS functions exists for ensuring the efficient operation of the system itself via resource sharing
  - **Resource allocation -** When  multiple users or multiple jobs running concurrently, resources must be allocated to each of them
    - Many types of resources -  Some (such as CPU cycles, main memory, and file storage) may have special allocation code, others (such as I/O devices) may have general request and release code
  - **Accounting -** To keep track of which users use how much and what kinds of computer resources
  - **Protection and security -** The owners of information stored in a multiuser or networked computer system may want to control use of that information, concurrent processes should not interfere with each other
    - **Protection** involves ensuring that all access to system resources is controlled
    - **Security** of the system from outsiders requires user authentication, extends to defending external I/O devices from invalid access attempts

# A View of Operating System Services



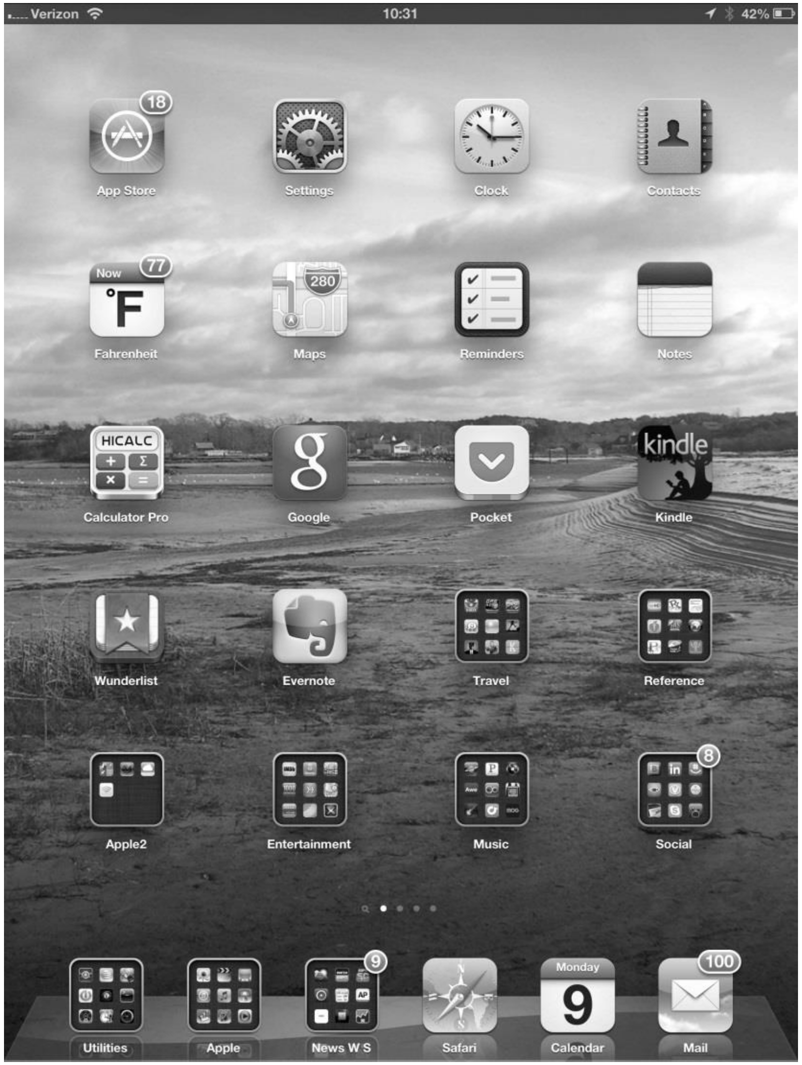
# User Operating System Interface - GUI

- User-friendly **desktop** metaphor interface
  - Usually mouse, keyboard, and monitor
  - **Icons** represent files, programs, actions, etc.
  - Various mouse buttons over objects in the interface cause various actions (provide information, options, execute function, open directory (known as a **folder**)
  - Invented at Xerox PARC
- Many systems now include both Command Line Interface (CLI) and GUI interfaces
  - Microsoft Windows is GUI with CLI "command" shell
  - Unix and Linux have CLI with optional GUI interfaces (CDE, KDE, GNOME)

# Touchscreen Interfaces

Touchscreen devices require new interfaces

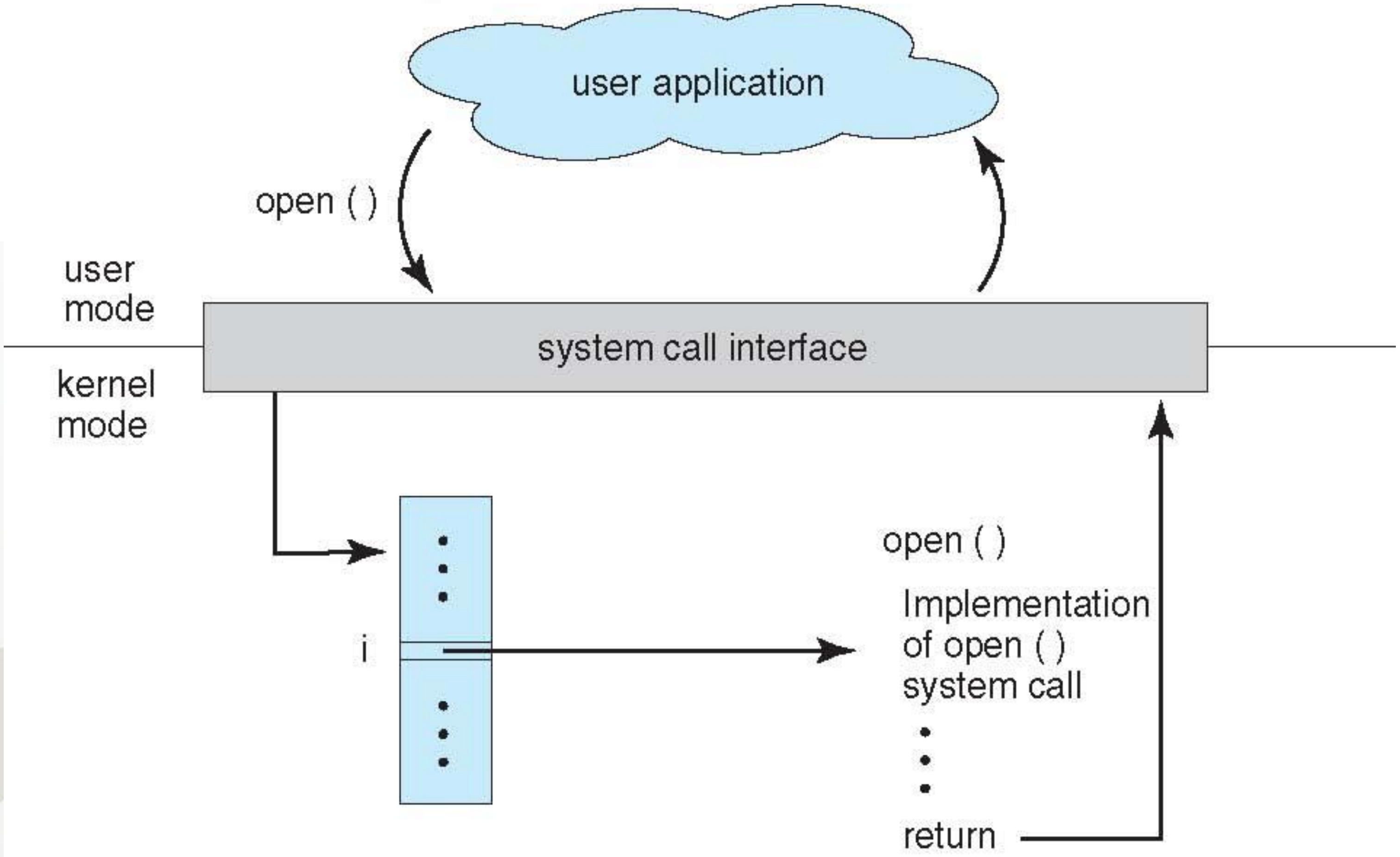
- Mouse not possible or not desired
- Actions and selection based on gestures
- Virtual keyboard for text entry



# System Calls

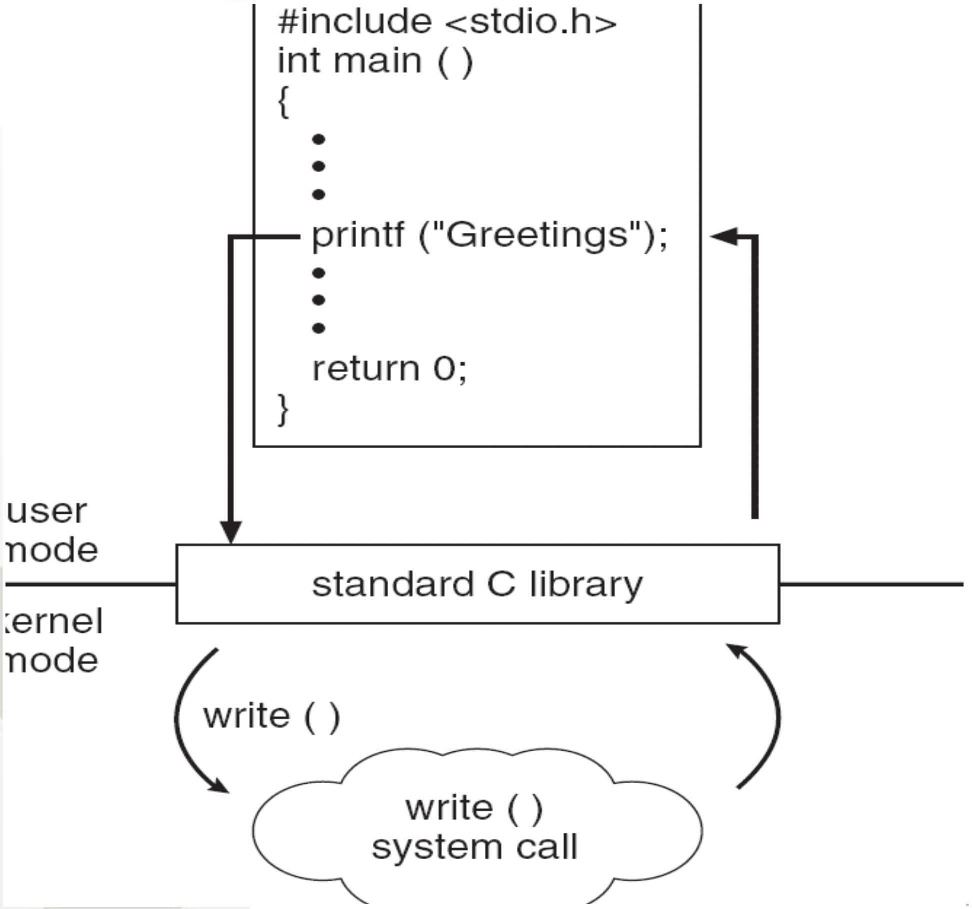
- **Programming interface to the services provided by the OS**
- Typically written in a high-level language (C or C++)
- Mostly accessed by programs via a high-level **Application Program Interface (API)** rather than direct system call use
- Three most common APIs are Win32 API for Windows, POSIX API for POSIX-based systems (including virtually all versions of UNIX, Linux, and Mac OS X), and Java API for the Java virtual machine (JVM)

# API – System Call – OS Relationship



# Standard C Library Example

- C program invoking printf() library call, which calls write() system call



```
#include <stdio.h>
int main ( )
{
   •
   •
   •
   printf ("Greetings");
   •
   •
   •
   return 0;
}
```

# Kernel Tasks

- Kernel - the central software that manages and allocates computer resources (i.e., the CPU, RAM, and devices).

1. Process scheduling:
    - Linux is a preemptive multitasking operating system, Multitasking means that multiple processes (i.e., running programs) can simultaneously reside in memory and each may receive use of the CPU(s).
2. Memory management:
3. Provision of a file system
4. Creation and termination of processes:
5. Access to devices
6. Networking:
7. Provision of a system call API

# System Boot

- Operating system must be made available to hardware, so hardware can start it
  - Small piece of code – **bootstrap loader**, locates the kernel, loads it into memory, and starts it
  - Sometimes two-step process where **boot block** at fixed location loads bootstrap loader
  - When power initialized on system, execution starts at a fixed memory location
    - Firmware used to hold initial boot code

# Process Concept

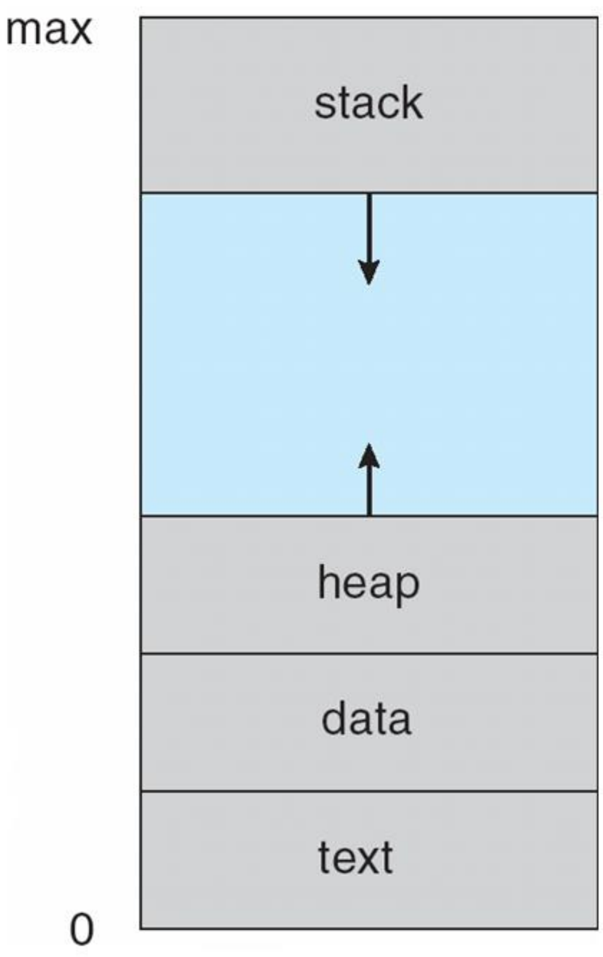
- An operating system executes a variety of programs:
  - Batch system – **jobs**
  - Time-shared systems – **user programs** or **tasks**

- Terms ***job*** and ***process*** are used almost interchangeably

- Program is ***passive*** entity stored on disk (**executable file**), process is ***active***
  - Program becomes process when executable file loaded into memory

- Execution of program started via GUI mouse clicks, command line entry of its name, etc.
- One program can be several processes
  - Consider multiple users executing the same program

# Process in Memory

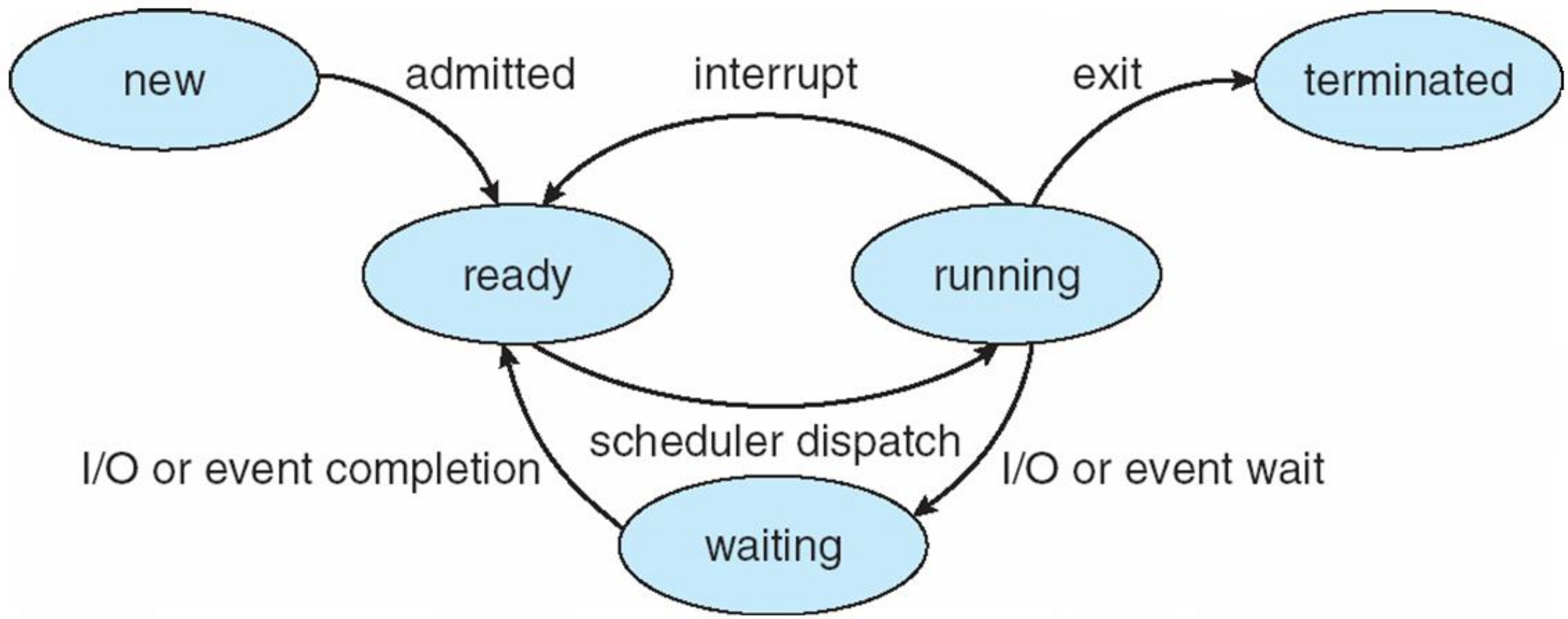


- **Process** – a program in execution;
- Process execution must progress in sequential fashion
- Multiple parts:
  - The program code, also called **text section**
  - Current activity including **program counter**, processor registers
  - **Stack** containing temporary data, function parameters, return addresses, local variables
  - **Data section** containing global variables
  - **Heap** containing memory dynamically allocated during run time

# Process State

- As a process executes, it changes **state**
  - **new:** The process is being created
  - **ready:** The process is waiting to be assigned to a processor
  - **running:** Instructions are being executed
  - **waiting:** The process is waiting for some event to occur
  - **terminated:** The process has finished execution

# Diagram of Process State

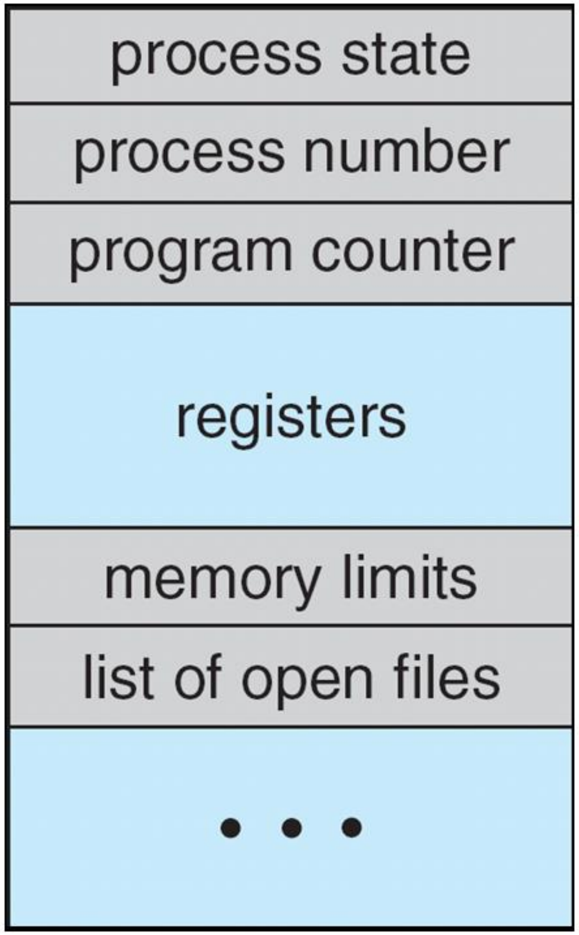


# Process Control Block (PCB)

Each process is represented by a **process control block** (also called **task control block**)

Contains information associated with each process

- **Process state** – running, waiting, etc.
- **Program counter** – location of instruction to next execute
- **CPU registers** – contents of all process-centric registers
- **CPU scheduling information-** priorities, scheduling queue pointers
- **Memory-management information** – memory allocated to the process
- **Accounting information** – CPU used, clock time elapsed since start, time limits
- **I/O status information** – I/O devices allocated to process, list of open files



process state

process number

program counter

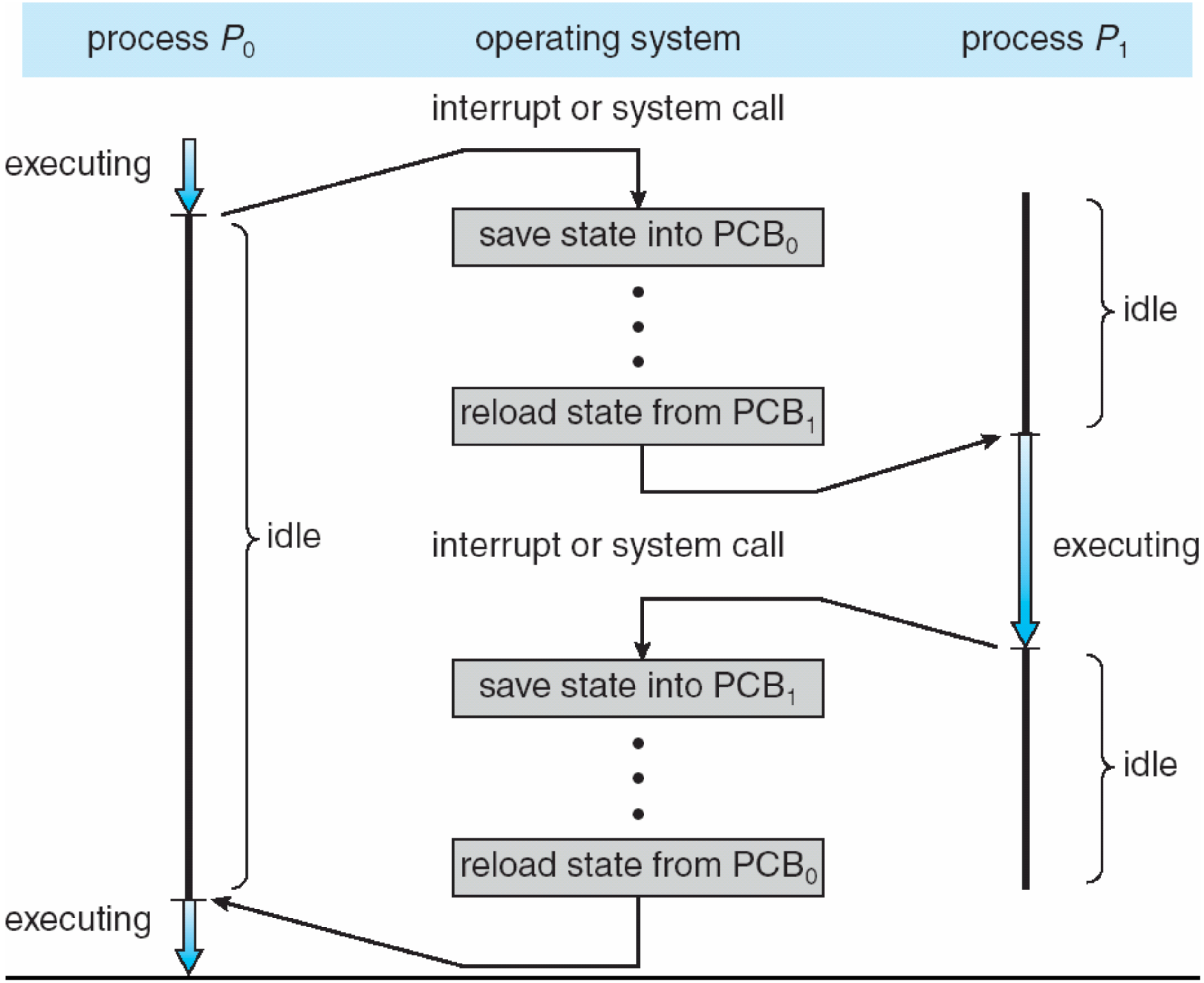
registers

memory limits

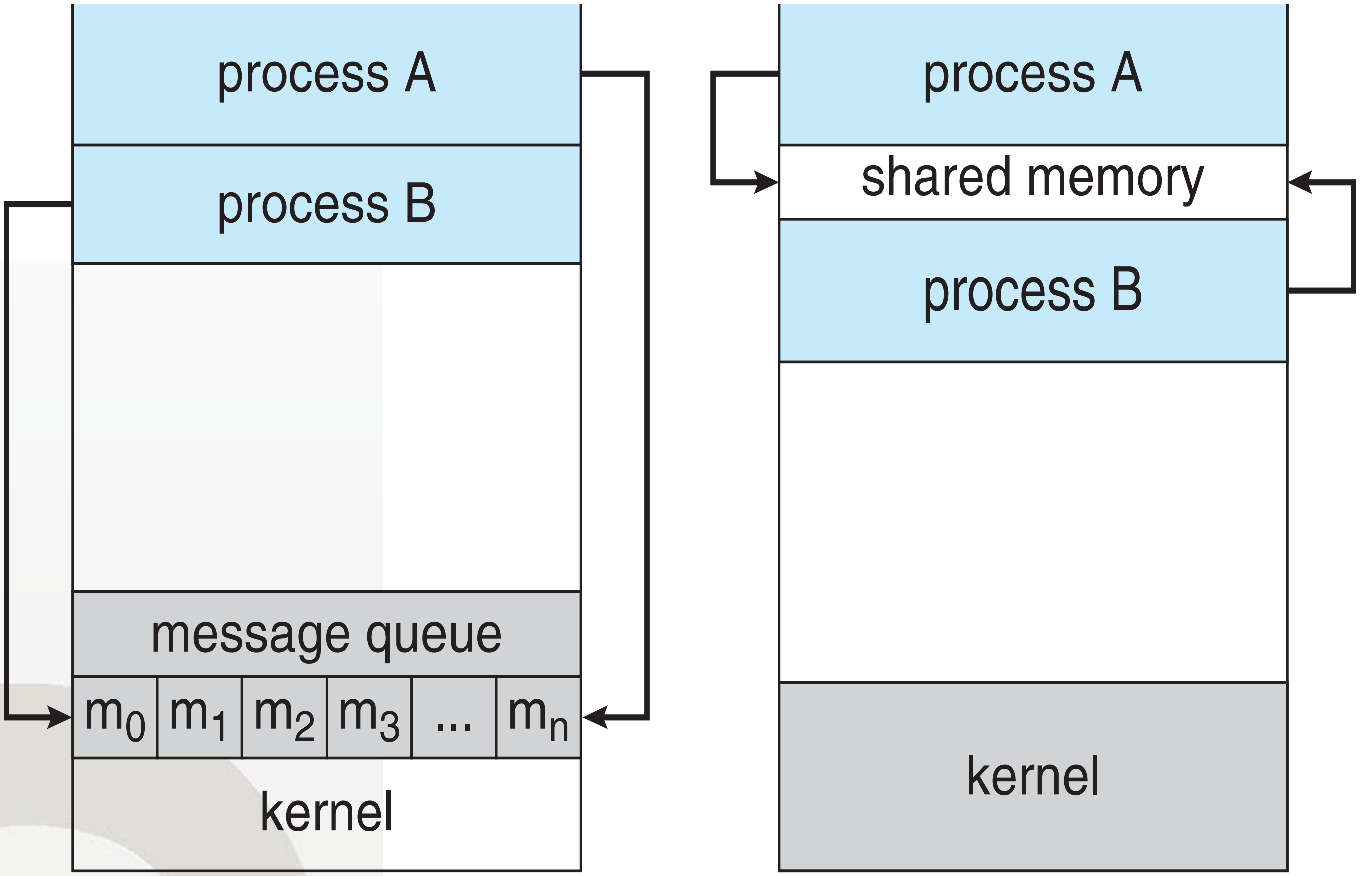
list of open files

• • •

# CPU Switch From Process to Process



# Interprocess Communication



(a)

Message Passing

(b)

Shared Memory

# Concurrency

- In real life we act on a number of concurrent stimuli, say reading and at the same time listening to music.
- The instrumentation in a car collects and display continous information about speed, fuel consumption etc, regulate the indoor temperature according to a set value, monitors the status of oil, brakes and whatever, controls the speed by means of the cruise control system.
- The ABS (Anti-Lock Braking System) prevents locking of a vehicle’s wheels during braking.
- All together, most measurement and control systems are inherently concurrent.

# True Concurrency vs. Pseudo-Concurrency

- Computer science defines concurrency as a property of systems where several processes are executing at the same time, and may or may not interact with each other.
- In a single processor, concurrent external activities must be ”mapped” as pseudo-concurrent sequential processes. The timing requirements is met when all the sequential processes can react within the given deadlines. This may be difficult to prove for hard Real-Time systems.
- True concurrency requires parallell processing in separate processors, either a multi-processor system or multi-CPUs
- However, in most cases several tasks can be executed pseduo-concurrently in a single processor;

# Real Time - Concurrent Systems

- A dedicated (embedded) system using a microcontroller to read out data at a fixed rate or from interrupts is a Real-time system, but is **not** a concurrent system if all processing is done within the same process
- Vice versa: a concurrent system where the correctness does not really depend on timing constraints, is **not** a Real Time system
- However, in general a Real -Time system is a concurrent system, where each Real Time activitiy is mapped into a process (or a thread, or a ”task”)

# Concurrent Programming

- Concurrent programming notation expresses potential parallelism and dealing with **synchronization** and **communication** problems
- For Real-Time systems the physical activities are mapped into a number of separately executing programs, processes
  - **Note the distinction between the program code and the execution of this code within the context of a process. The same code can be executed by several processes, operating on separate data.**
- All Real-Time systems are inherently concurrent — physical devices operate in **parallel** in the real world

# Execution of Real-Time Processes – How?

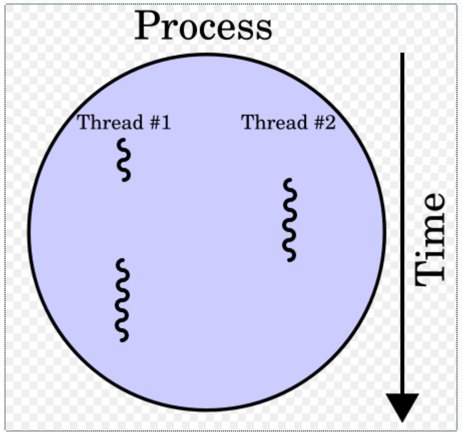
- A process can be executed, or resumed after a wait, by:
    - User/operator command
    - Data **interrupt**, for instance from the ADC
    - By a clock (timer), typical for a **periodic** execution
        - Timer interrupt
        - On a given date and time
    - By a **signal** or **message** from another process
- If several processes want to execute at the same time, a mechanism is required to select which process shall get the CPU

# Process vs. Threads

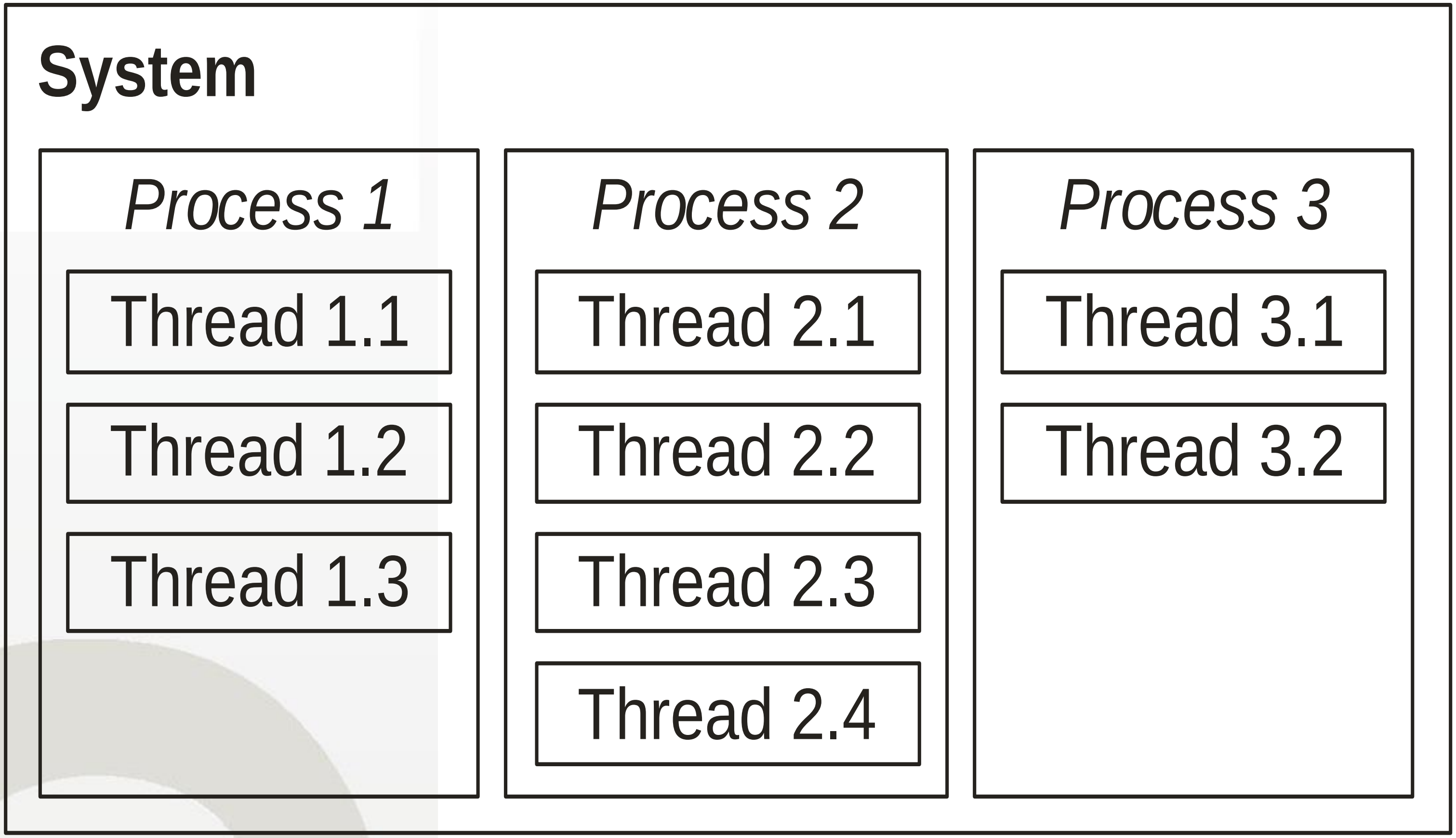
**A process** (synonymously called "task") is

- An abstraction of a running program
- The logical unit of work schedulable by the OS

**A thread** is a lightweight process within a regular process. It has access to the same memory space, and the context switching from one thread to another will be shorter than for process to process



# Processes, and Multiple Threads



System

Process 1

Thread 1.1

Thread 1.2

Thread 1.3

Process 2

Thread 2.1

Thread 2.2

Thread 2.3

Thread 2.4

Process 3

Thread 3.1

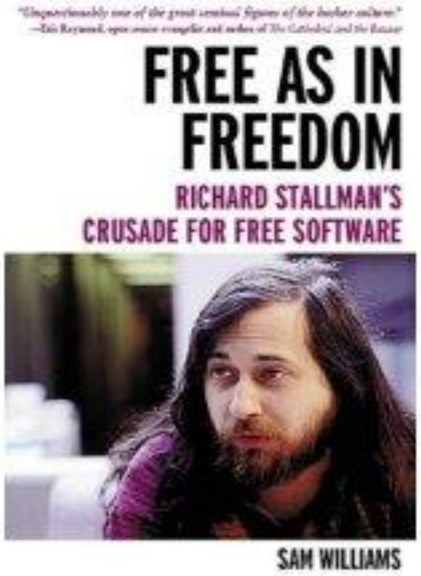
Thread 3.2

# Before Linux

- In 80’s, Microsoft’s DOS was the dominated OS for PC
- Apple MAC was better, but expensive
- UNIX was much better, but much, much more expensive. Only for minicomputer for commercial applications
- People was looking for a UNIX based system, which is cheaper and can run on PC
- Both DOS, MAC and UNIX were proprietary, i.e., the source code of their kernel is protected
- No modification is possible without paying high license fees

# GNU project

- Established in 1984 by Richard Stallman, who believes that software should be free from restrictions against copying or modification in order to make better and efficient computer programs
- Aim to create a free UNIX like OS, consisting of a kernel and all associated software packages


"Unquestionably one of the great seminal figures of the hacker culture."
—Eric Raymond, open source evangelist and author of The Cathedral and the Bazaar
FREE AS IN FREEDOM
RICHARD STALLMAN'S CRUSADE FOR FREE SOFTWARE
SAM WILLIAMS

GNU is a recursive acronym for "GNU's Not Unix"

Aim at developing a complete Unix-like operating system which is free for copying and modification

Companies make their money by maintaining and distributing the software, e.g. optimally packaging the software with different tools (Redhat, Slackware, Mandrake, SuSE, etc)

Stallman built the first free GNU C Compiler in 1991. But still, an OS was yet to be developed

# Beginning of Linux

- A famous Holland professor from Andrew Tanenbaum developed Minix, a simplified version of UNIX that runs on PC in mid-1980s
- Minix is for class teaching only. No intention for commercial use
- Inspired by Minux in Sept 1991, Linus Torvalds, a second year student of Computer Science at the University of Helsinki, developed the preliminary kernel of Linux, known as Linux version 0.0.1 for running on his Intel 80386 PC

# Message from Professor Andrew Tanenbaum

**"I still maintain the point that designing a monolithic kernel in 1991 is a fundamental error.  Be thankful you are not my student.  You would not get a high grade for such a design :-)“**

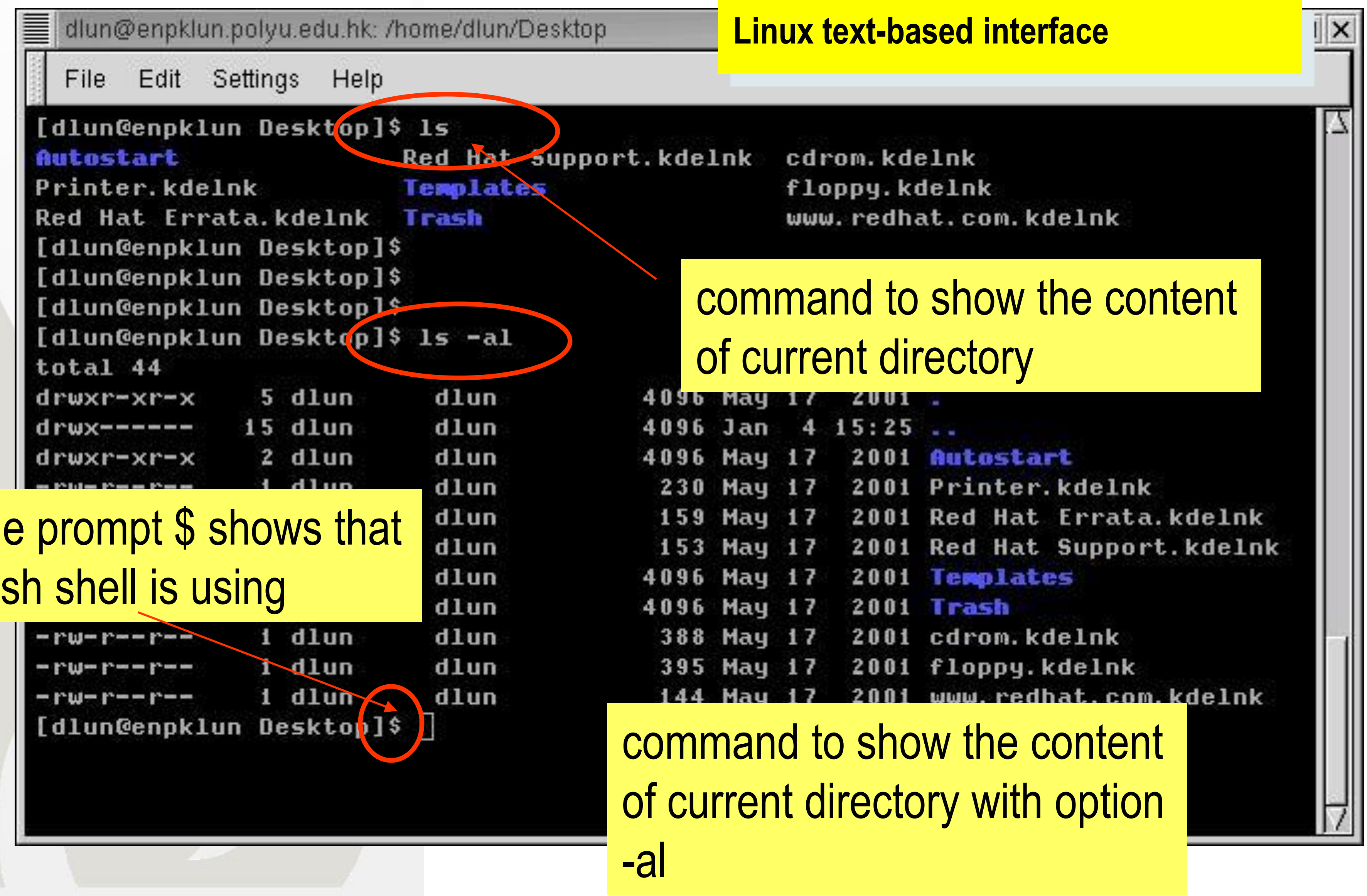
(Andrew Tanenbaum to Linus Torvalds)

- Soon more than a hundred people joined the Linux camp. Then thousands. Then hundreds of thousands
- It was licensed under GNU General Public License, thus ensuring that the source codes will be free for all to copy, study and to change.
- Latest stable release is Linux 14.10.7 from August 2013

# Linux Today

- Linux has been used for many computing platforms
  - PC, PDA, Supercomputer,…
- Not only character user interface but graphical user interface is available
- Commercial vendors moved in Linux itself to provide freely distributed code. They make their money by compiling up various software and gathering them in a distributable format
  - Red Hat, RTLinux, etc

All LINUX commands start with the name of the command and can be followed by options and arguments.



```
[dlun@enpklun Desktop]$ ls
Autostart              Red Hat Support.kdelnk  cdrom.kdelnk
Printer.kdelnk         Templates               floppy.kdelnk
Red Hat Errata.kdelnk  Trash                   www.redhat.com.kdelnk
[dlun@enpklun Desktop]$
[dlun@enpklun Desktop]$
[dlun@enpklun Desktop]$
[dlun@enpklun Desktop]$ ls -al
total 44
drwxr-xr-x    5 dlun     dlun         4096 May 17  2001 .
drwx------   15 dlun     dlun         4096 Jan  4 15:25 ..
drwxr-xr-x    2 dlun     dlun         4096 May 17  2001 Autostart
-rw-r--r--    1 dlun     dlun          230 May 17  2001 Printer.kdelnk
                         dlun          159 May 17  2001 Red Hat Errata.kdelnk
                         dlun          153 May 17  2001 Red Hat Support.kdelnk
                         dlun         4096 May 17  2001 Templates
                         dlun         4096 May 17  2001 Trash
-rw-r--r--    1 dlun     dlun          388 May 17  2001 cdrom.kdelnk
-rw-r--r--    1 dlun     dlun          395 May 17  2001 floppy.kdelnk
-rw-r--r--    1 dlun     dlun          144 May 17  2001 www.redhat.com.kdelnk
[dlun@enpklun Desktop]$
```

**Linux text-based interface**

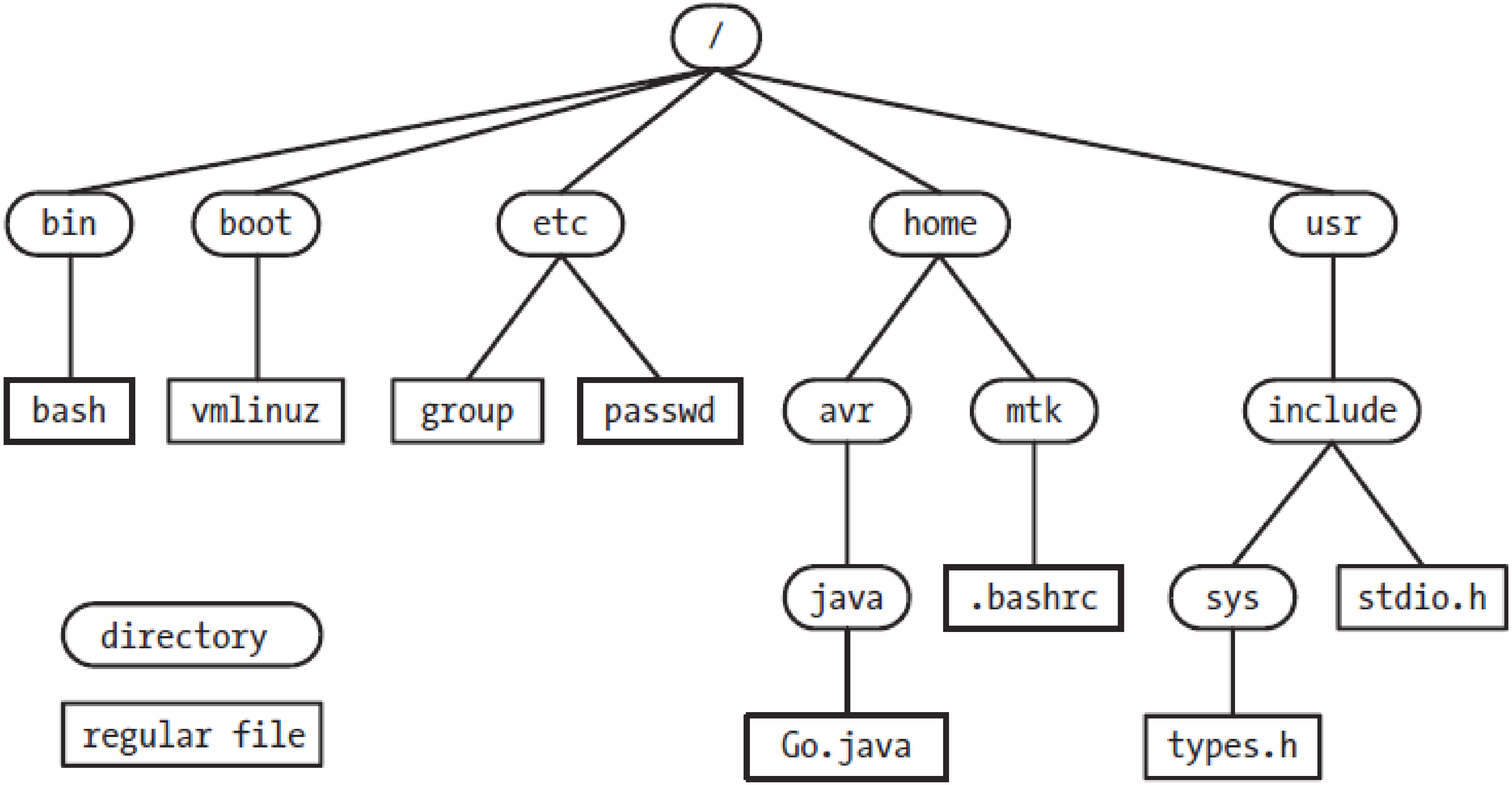
command to show the content of current directory

The prompt $ shows that bash shell is using

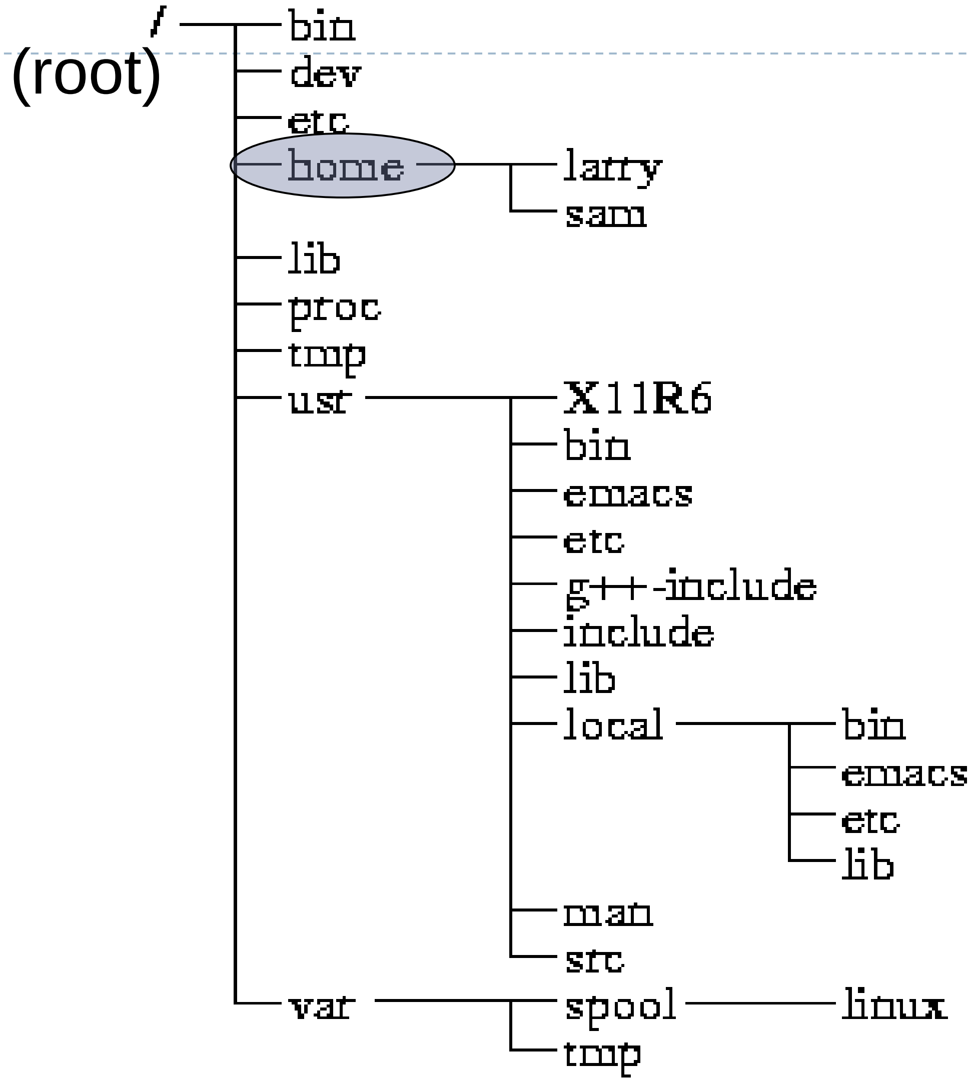
command to show the content of current directory with option -al

# Linux Single Directory Hierarchy

The Linux kernel maintains a single hierarchical directory structure to organize all files in the system.



# Directory Tree



When you log on the the Linux OS using your username you are automatically located in your home directory.

# The most important subdirectories inside the root directory are:

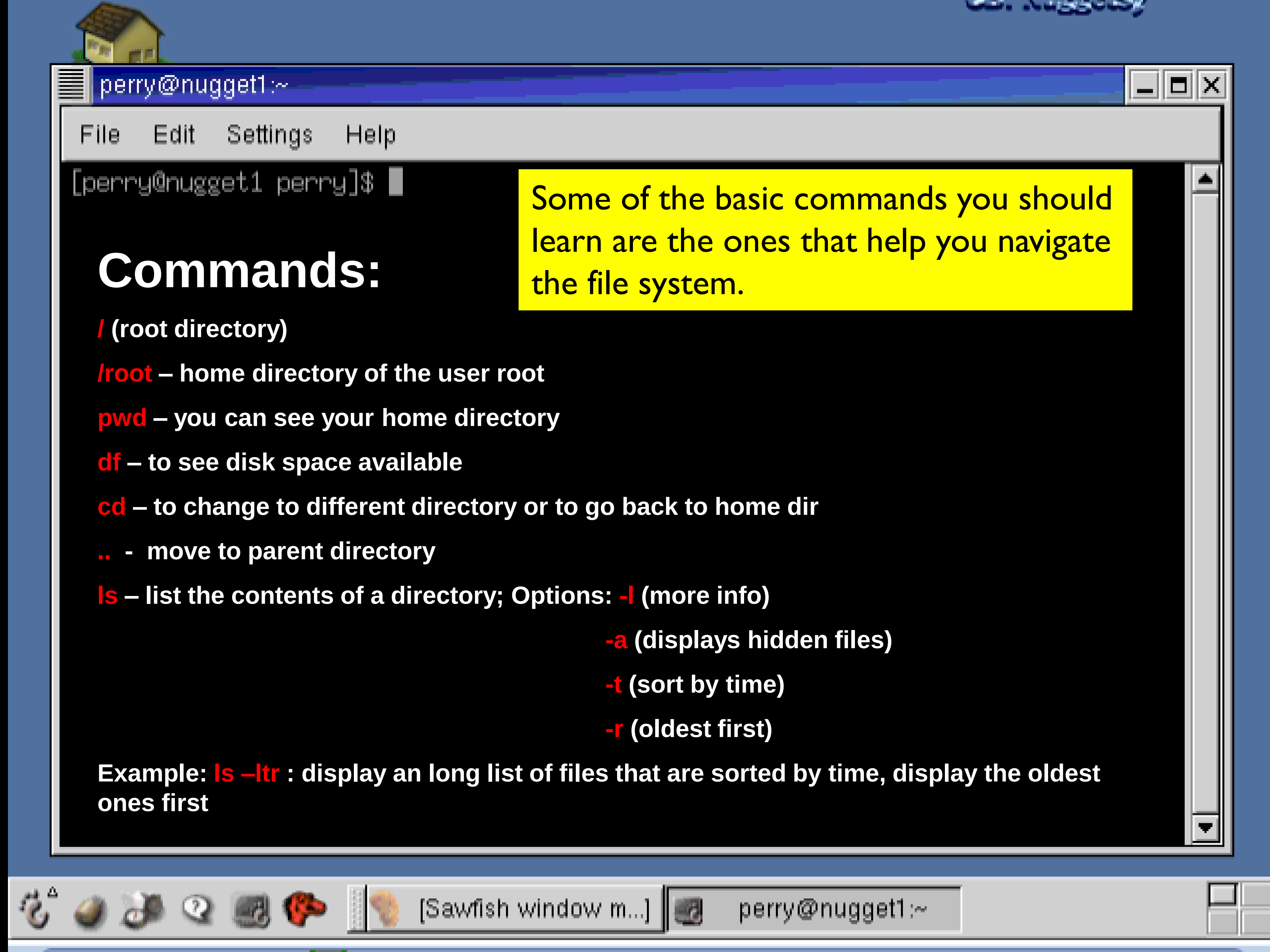
- **/bin** : Important Linux commands available to the average user.
- **/boot** : The files necessary for the system to boot. Not all Linux distributions use this one. Fedora does.
- **/dev** : All device drivers. Device drivers are the files that your Linux system uses to talk to your hardware. For example, there's a file in the /dev directory for your particular make and model of monitor, and all of your Linux computer's communications with the monitor go through that file.
- **/etc** : System configuration files.
- **/home** : Every user except root gets her own folder in here, named for her login account. So, the user who logs in with linda has the directory /home/linda, where all of her personal files are kept.
- **/lib** : System libraries. Libraries are just bunches of programming code that the programs on your system use to get things done.

# The most important subdirectories inside the root directory are:

- **/mnt :** Mount points. When you temporarily load the contents of a CD-ROM or USB drive, you typically use a special name under /mnt. For example, many distributions (including Fedora) come, by default, with the directory /mnt/cdrom, which is where your CD-ROM drive's contents are made accessible.
- **/root** : The root user's home directory.
- **/sbin** : Essential commands that are only for the system administrator.
- **/tmp** : Temporary files and storage space. Don't put anything in here that you want to keep. Most Linux distributions (including Fedora) are set up to delete any file that's been in this directory longer than three days.
- **/usr** : Programs and data that can be shared across many systems and don't need to be changed.
- **/var** : Data that changes constantly (log files that contain information about what's happening on your system, data on its way to the printer, and so on).

# Home directory

- You can see what your home directory is called by entering
- pwd (print current working directory)



```
[perry@nugget1 perry]$ 
```

Some of the basic commands you should learn are the ones that help you navigate the file system.

# Commands:

**/ (root directory)**

**/root – home directory of the user root**

**pwd – you can see your home directory**

**df – to see disk space available**

**cd – to change to different directory or to go back to home dir**

**.. - move to parent directory**

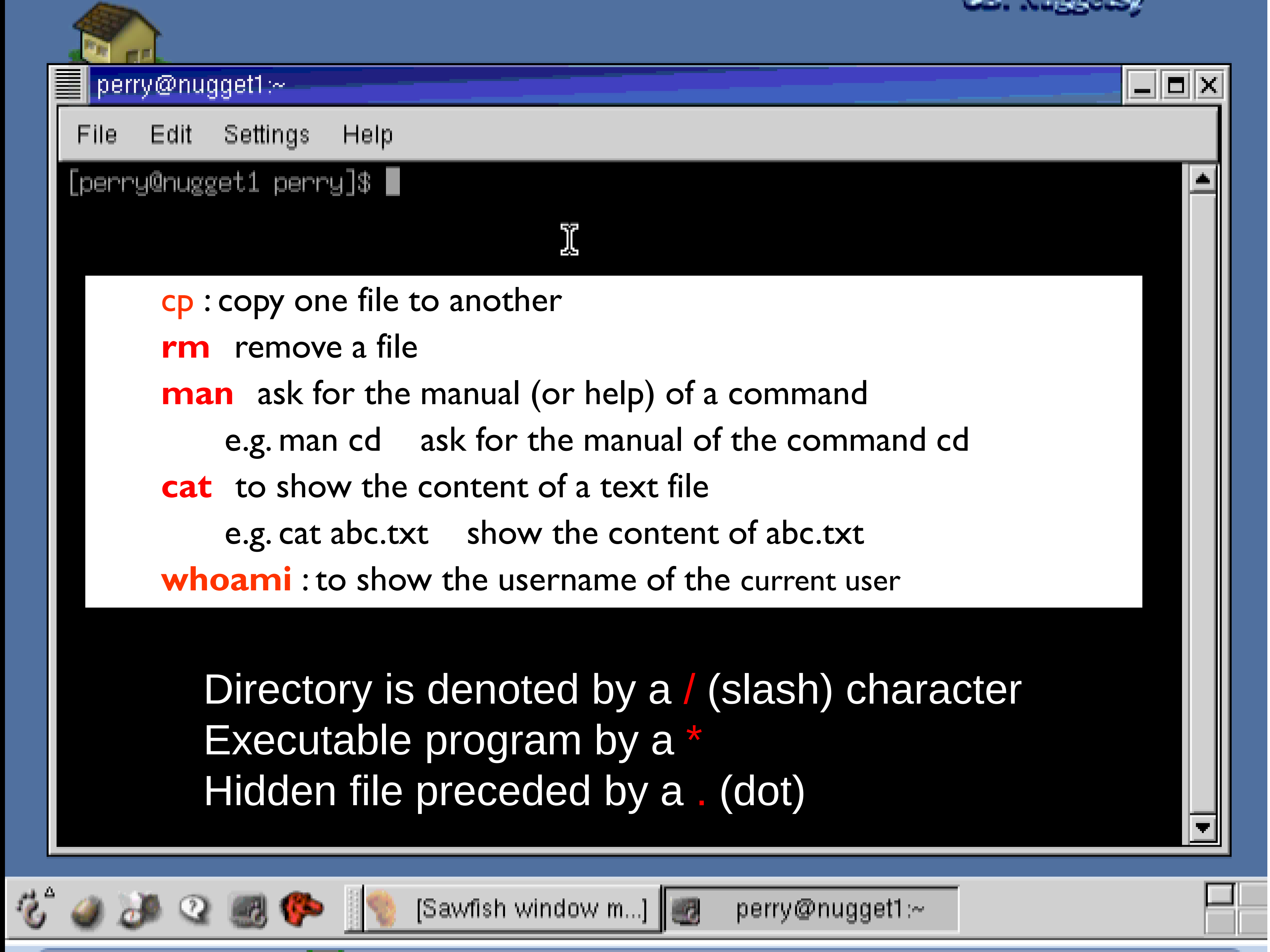
**ls – list the contents of a directory; Options: -l (more info)**

**-a (displays hidden files)**

**-t (sort by time)**

**-r (oldest first)**

**Example: ls –ltr : display an long list of files that are sorted by time, display the oldest ones first**



```
[perry@nugget1 perry]$
```

**cp** : copy one file to another

**rm** remove a file

**man** ask for the manual (or help) of a command

e.g. man cd ask for the manual of the command cd

**cat** to show the content of a text file

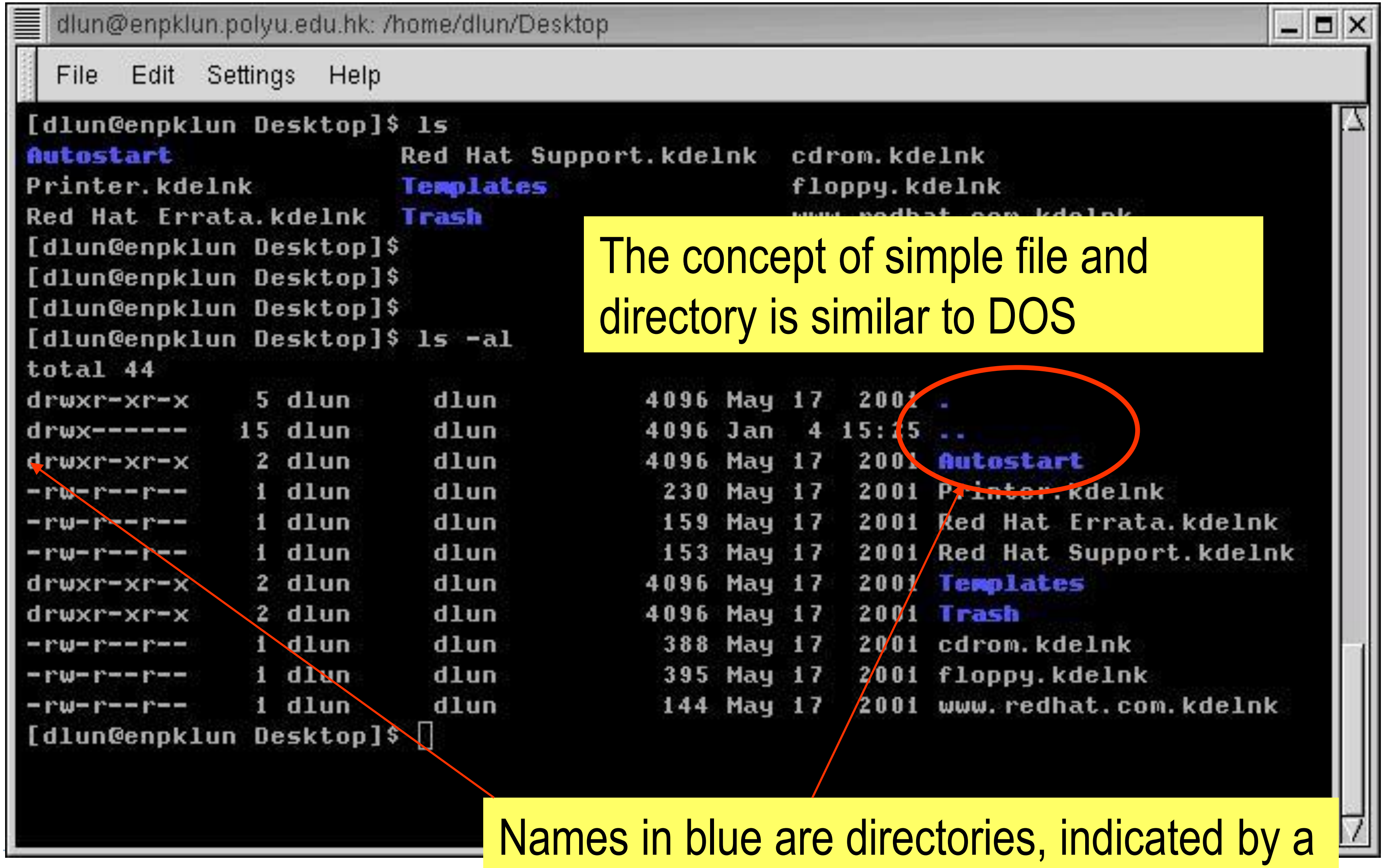
e.g. cat abc.txt show the content of abc.txt

**whoami** : to show the username of the current user

Directory is denoted by a / (slash) character

Executable program by a *

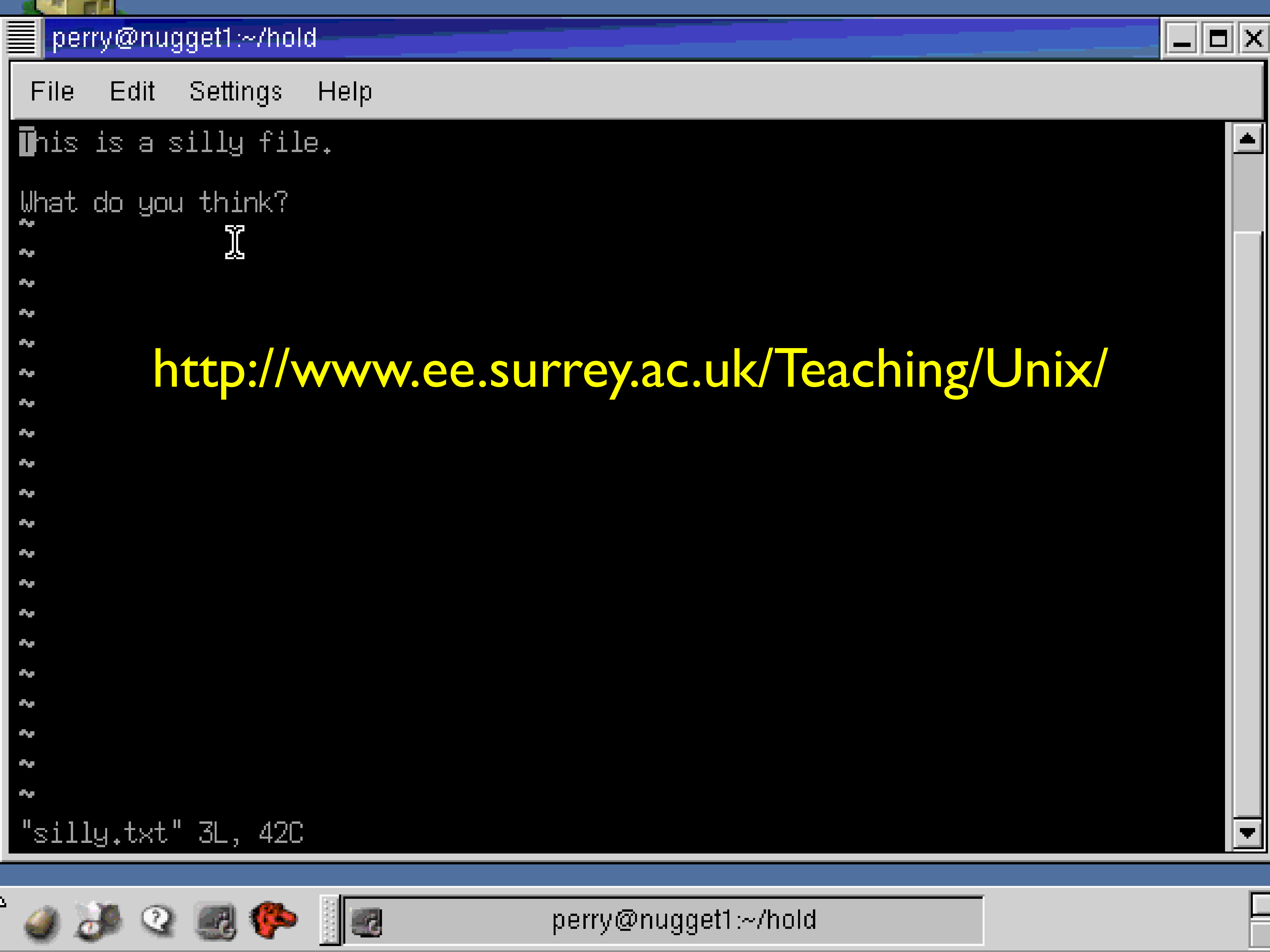
Hidden file preceded by a . (dot)



```
[dlun@enpklun Desktop]$ ls
Autostart              Red Hat Support.kdelnk   cdrom.kdelnk
Printer.kdelnk         Templates                floppy.kdelnk
Red Hat Errata.kdelnk  Trash                    www.redhat.com.kdelnk
[dlun@enpklun Desktop]$
[dlun@enpklun Desktop]$
[dlun@enpklun Desktop]$
[dlun@enpklun Desktop]$ ls -al
total 44
drwxr-xr-x    5 dlun     dlun         4096 May 17  2001 .
drwx------   15 dlun     dlun         4096 Jan  4 15:25 ..
drwxr-xr-x    2 dlun     dlun         4096 May 17  2001 Autostart
-rw-r--r--    1 dlun     dlun          230 May 17  2001 Printer.kdelnk
-rw-r--r--    1 dlun     dlun          159 May 17  2001 Red Hat Errata.kdelnk
-rw-r--r--    1 dlun     dlun          153 May 17  2001 Red Hat Support.kdelnk
drwxr-xr-x    2 dlun     dlun         4096 May 17  2001 Templates
drwxr-xr-x    2 dlun     dlun         4096 May 17  2001 Trash
-rw-r--r--    1 dlun     dlun          388 May 17  2001 cdrom.kdelnk
-rw-r--r--    1 dlun     dlun          395 May 17  2001 floppy.kdelnk
-rw-r--r--    1 dlun     dlun          144 May 17  2001 www.redhat.com.kdelnk
[dlun@enpklun Desktop]$
```

The concept of simple file and directory is similar to DOS

Names in blue are directories, indicated by a letter d at the beginning of the line



```
This is a silly file.

What do you think?
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"silly.txt" 3L, 42C
```

http://www.ee.surrey.ac.uk/Teaching/Unix/

# Any Questions?