Operating System

Lecture 1,2,3: Course aims, computer system, intro to os, os abstract view, program run, functions and goals of os

By Prof K R Chowdhary

JNV University

2023

Course Aims

- This course aims to:
  - explain the structure and functions of an operating system,
  - illustrate key operating system aspects by concrete example, and
  - prepare you for future courses. . .
- At the end of the course you should be able to:
  - compare and contrast CPU scheduling algorithms
  - explain the following: process, address space, file.
  - distinguish paged and segmented virtual memory.
  - discuss the relative merits of Unix and NT. .

Course outlines

- Introduction to Operating Systems.
- Processes & Scheduling.
- Memory Management.
- I/O & Device Management.
- Protection.
- Filing Systems.
- Case Study: Unix.
- Case Study: Windows

Computer System

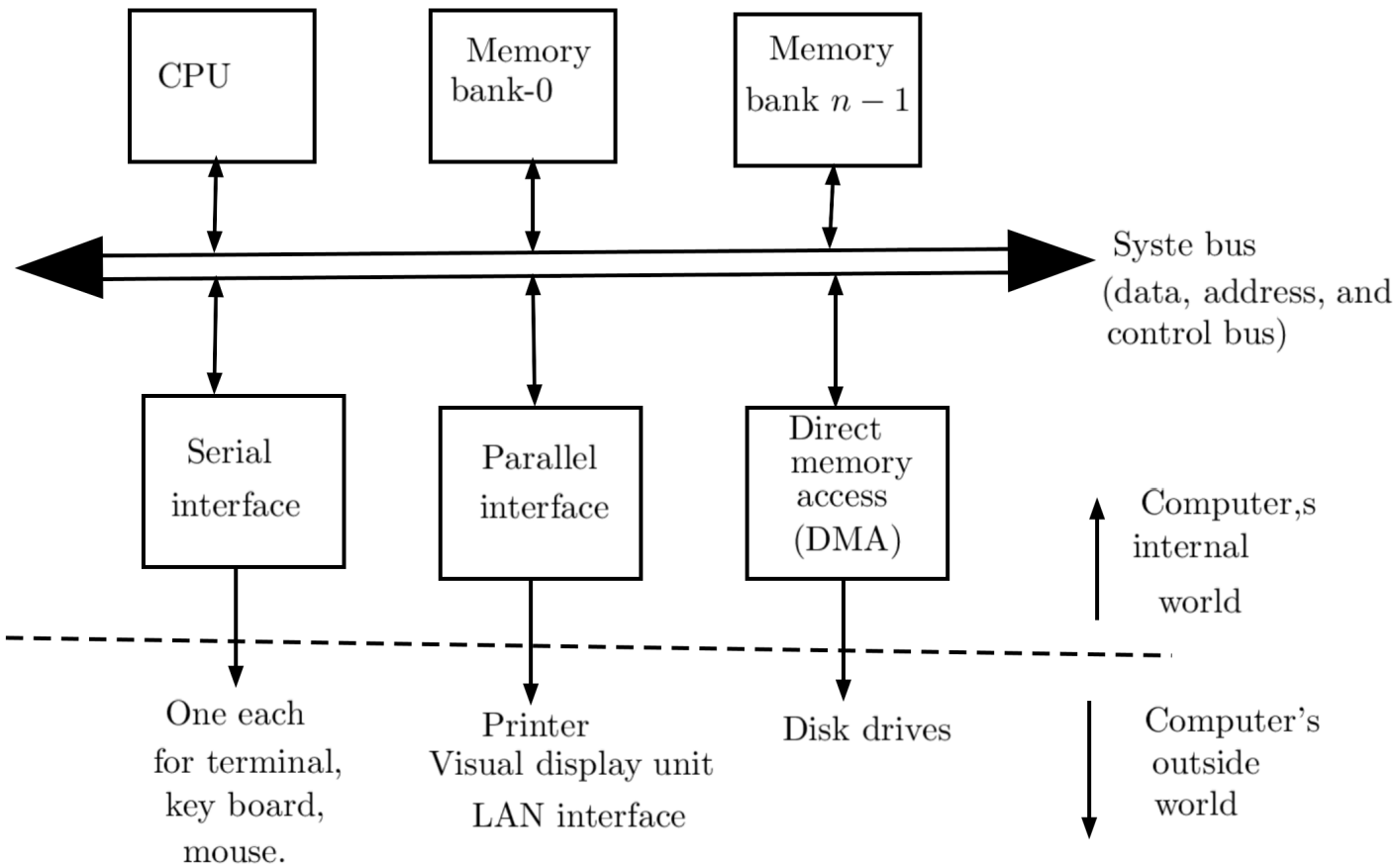


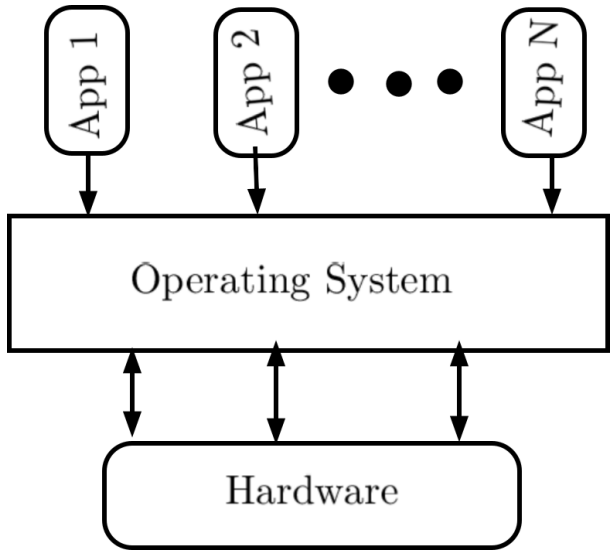
Figure 1: Computer System

Introduction to Operating System

What is an operating system?

- A program which controls the execution of all other programs (applications).
- Middleware between user programs and system hardware
- Manages hardware: CPU, main memory, IO devices (monitor, disk, network card, mouse, keyboard etc.)
- Objectives:
  - convenience,
  - efficiency,
  - extensibility.
- Similar to government ....

An abstract view of OS



- The Operating System (OS):
  - controls all execution.
  - multiplexes resources between applications.
  - abstracts away from complexity.

Figure 2: AN abstract view of OS

- Typically also have some libraries and some tools provided with OS.
- Are these part of the OS? Is *browser* a tool?

  no-one can agree...
- For unix/linux user, the OS = the kernel.

What happens when you run a program? (Background)

- A compiler translates high level programs into an executable (".c" to "a.out")
- The exe contains instructions that the CPU can understand, and data of the program (all numbered with addresses)
- Instructions run on CPU: hardware implements an instruction set architecture (ISA)
- CPU also consists of a few registers, e.g.,
  - Pointer to current instruction (program counter or PC)
  - Operands of instructions, memory addresses (in general purpose registers)
  - One or more accumulators

What happens when you run a program?

- To run an exe (.exe file in windows), the CPU
  - fetches instruction pointed at by PC from memory into IR (instruction register)
  - loads data required by the instructions into registers
  - decodes and executes the instruction
  - stores results to memory
- Most recently used instructions and data are in CPU are loaded in caches for faster access

What does the OS do?

- OS manages CPU
  - Initializes program counter (PC) and other registers to begin execution
- OS manages program memory
  - Loads program executable (code, data) from disk to memory
- OS manages external devices
  - Read/write files from disk.

1. OS manages CPU

- OS provides the process abstraction
  - Process: a running program
  - OS creates and manages processes
- Each process has the illusion of having the complete CPU, i.e., OS virtualizes CPU
- Timeshares CPU between processes
- Enables coordination between processes

2. OS manages memory

- OS manages the memory of the process: code, data, stack, heap etc
- Each process thinks it has a dedicated memory space for itself, numbers code and data starting from 0 (virtual addresses)
- OS abstracts out the details of the actual placement in memory, translates from virtual addresses to actual physical addresses

2. OS manages memory...

This program allocates memory, inputs text string followed by ‘0’ and stores in allocated memory and prints that on pressing of EOF char (ctrl-d).

krc@krc-Inspiron-13-5378: ~/works/operating-system/my_os...

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 int main(){
4         char ch, *hello;
5         int i=0;
6         hello = malloc(sizeof(char)*200);
7         while((ch=getchar())!=EOF){
8                 hello[i]=ch;
9                 i++;
10        }
11        hello[i]='\0';
12        printf("This is what you have input and stored in RAM\n");
13        i=0;
14        while(hello[i]!='\0'){
15                putchar(hello[i]);
16                i++;
17        }
18 free(hello);
19 return 0;
20 }
                                                    11,14-21      All
```

2. OS manages memory...

Running of above program:

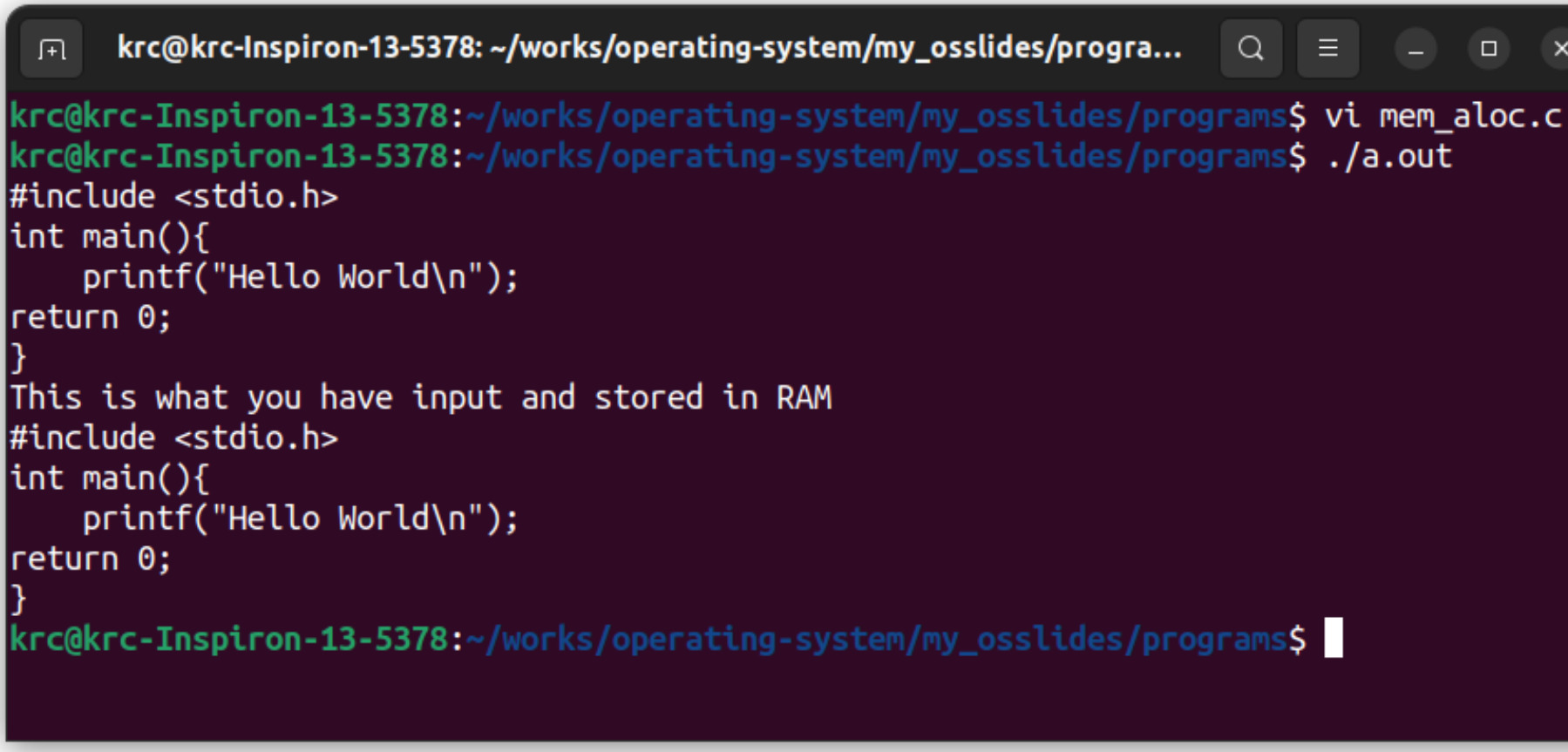


Figure 4: Running of program having "system calls": memalloc() and free()

3. OS manages devices

- OS has code to manage disk, network card, and other external devices: device drivers
- Device driver talks the language of the hardware devices
  - Issues instructions to devices (fetch data from a file)
  - Responds to interrupt events from devices (user has pressed a key on keyboard)
- Persistent data organized as a filesystem on disk

Design goals of an operating system

- Convenience, abstraction of hardware resources for user programs
- Efficiency of usage of CPU, memory, etc.
- Issues instructions to devices (fetch data from a file)
- Isolation between multiple processes

History of operating systems

- Started out as a library to provide common functionality across programs
- Later, evolved from procedure call to system call: what's the difference?
- When a system call is made to run OS code, the CPU executes at a *higher privilege* level
- Evolved from running a single program to multiple processes concurrently