Process

Hui Chen [a]

[a]CUNY Brooklyn College

Feburary 26, 2020

Outline

Outline
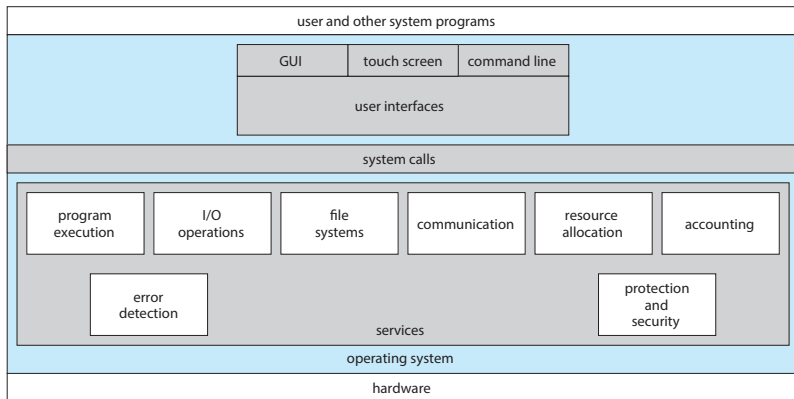
Overview of OS Services

Figure: A view of operating system services[1].

---

[1]Abraham Silberschatz, Peter B. Galvin, and Greg Gagne. *Operating system concepts*. 10th edition. John Wiley & Sons, 2018.

Evolved to *Process*

- ▶ Early computers run a single program at a time.
- ▶ Contemporary computer systems allow multiple programs to be loaded into memory and executed concurrently.
- ▶ This evolution results in the notation of a process, i.e., a program in execution.

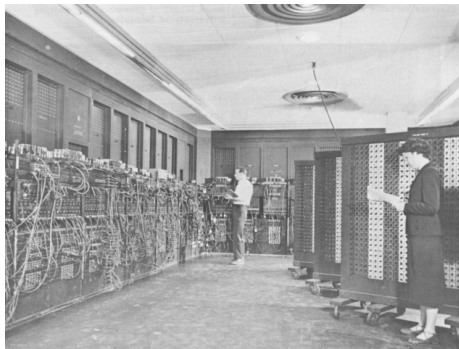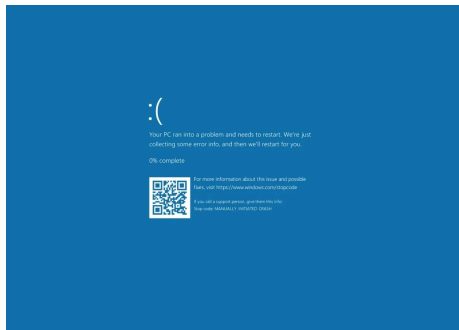Figure: From US Army Photo Archive

Process Management

- Why?
- What?
- How?

Outline

Need for Realiability and Protection

- ▶ Programs can fail.
- ▶ Programs may access other programs' code and data, intentionally or unintetionally.
- ▶ ...

Multiprogramming

- ▶ A technique in OS that OS organizes a collection of processes in such a way that the processes run concurrently (in parallel or in pseudo-parallel) and the CPU always has a process to execute.
- ▶ Multiprogramming can increase CPU utilization.

Time-Sharing

▶ A technique in OS that OS used a timer and cycle processes rapidly through the CPU , giving each user a share of the resources.

▶ Time-sharing can decrease response time to make system more responsive.

A Simple Model of Multiprogramming

$$\text{CPU Utilization} = 1 - p^n \tag{1}$$

where

- n. There are n processes in memory, and we call it the degree of multiprogramming.
- p. The fraction of time that a process spends in waiting for I/O to complete.

Outline

Policy

▶ Policy (What do we want?). The OS should provide protected access to shared resources.
  ▶ No program can read or write memory of another program or of the OS.
  ▶ Programs can run concurrently to utilize resources.
  ▶ *Additionally*, do we wish to have better utilization of system resources, do we wish to have better (small) response time?

Mechanism

▶ Policy (What?). The OS should provide protected access to shared resources.

▶ Mechanism (How?). Hardware, OS kernel, and *process*.

▶ Dual mode operating. Processes run in user or kernel mode. Restrict process in user mode from accessing privileged instructions.

▶ Address translation. Each process has its isolated logical (or virtual address space), and hardware translates virtual address to physical address.

Outline

Process in Memory

OS creates an illusion that a process runs as if the process ran on its own CPU and had its own memory, i.e., a process gets its own

- ▶ "virtual CPU", and
- ▶ "virtual memory address space"

Layout of Process in Memory

The layout of a process in memory consists of multiple parts, generally,

- ▶ Text. The text section is the program code.
- ▶ Stack. The stack section contains temporary data, such as, function parameters, return addresses, local variables.
- ▶ Data. The data section containing global variables
- ▶ Heap. The heap section containing memory dynamically allocated during run time.

Memory Layout of a C Program

Let's take a look at a C program ...

Multiple Processes in Memory

Consider that there are now multiple processes in memory.

▶ But there is only one single physical memory address space in the machine, how do we create the *illusion* that each has its own memory address space?

Address Translation

A process operates in an memory address space distinct from the physical memory space of the machine

▶ Address translation. Hardware translates a memory address the process operates on to a memory address in the physical memory.

Recall Discussion: Kernel and Process

▶ Policy. The OS should provide protected access to shared resources efficiently.
   ▶ No program can read or write memory of another program or of the OS.
   ▶ Programs can run concurrently to utilize resources efficiently.
▶ Mechanism. Hardware, OS kernel, and process.
   ▶ Dual mode operating. Processes run in user or kernel mode. Restrict process in user mode from accessing privileged instructions.
   ▶ Address translation. Each process has its isolated logical (or virtual address space), and hardware translates virtual address to physical address.
▶ *Experiment with an example buggy program to illustrate memory protection.*

Outline

Running Multiple Processes?

Consider that two processes run concurrently (in pseudo-parallel) in a single CPU core system ...

▶ These two processes switch on and off the CPU for many rounds ... how do we create the *illusion* that each has its own CPU?

Two processes are running ...

Consider

▶ At T_1, Process P_1 is active (on the physical CPU), and Process P_2 is inactive.

▶ At T_2, Process P_2 is active (on the physical CPU), and Process P_1 is inactive.

Two processes are running ...

How?

1. OS maintains a data structure in memory (its own address space), when the OS runs, it switches P_1 off the CPU and P_2 on the CPU, i.e., it

2. saves registers Program Counter (PC) and Stack Pointer (SP) in the data structure for P_1 in memory

3. loads PC and SP from the data structure for P_2 in memory

Process Control Block

A process control block (also called task control block or thread control block) is a data structure for information associated with each process including the process's execution context.

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

Outline

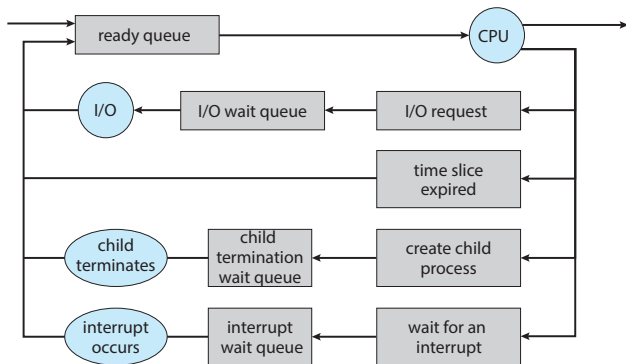# Process Queues and Transition

Figure: Process queues and transitions[2].

---

[2]Abraham Silberschatz, Peter B. Galvin, and Greg Gagne. *Operating system concepts*. 10th edition. John Wiley & Sons, 2018.
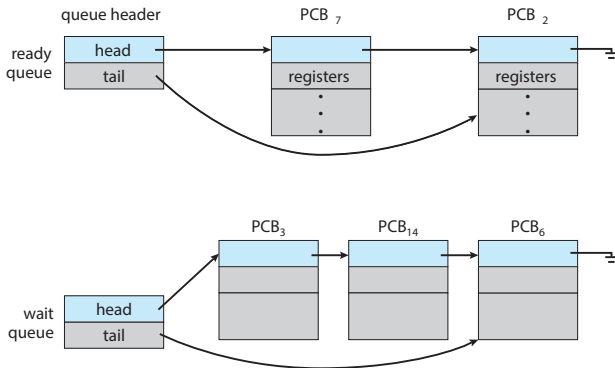
Data Structures for Process Queues

Figure: The *ready* and *wait* queues[3].

---

[3]Abraham Silberschatz, Peter B. Galvin, and Greg Gagne. *Operating system concepts*. 10th edition. John Wiley & Sons, 2018.
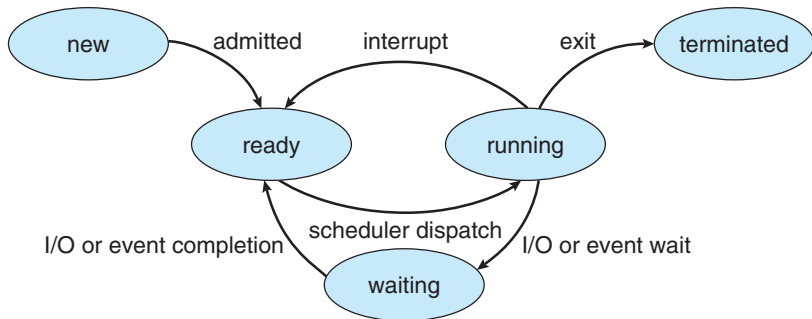
Process States and Transition

Figure: Process transitions in an OS[4].

---

[4]Abraham Silberschatz, Peter B. Galvin, and Greg Gagne. *Operating system concepts*. 10th edition. John Wiley & Sons, 2018.

Outline

Multiprogramming and Time-Sharing

▶ The objective of multiprogramming is to have some process running at all times so as to maximize CPU utilization.

▶ The objective of time sharing is to switch a CPU core among processes so frequently that users can interact with each program while it is running.

Questions. How do we support these objectives?

CPU Scheduler

▶ The CPU scheduler is to select from among the processes that are in the ready queue and allocate a CPU core to one of them.
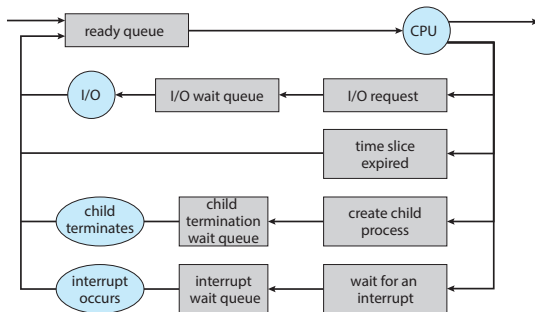▶ Different CPU scheduling algorithms help OSes meet different objectives.

Figure: Process queues and transitions[5].

---

[5]Abraham Silberschatz, Peter B. Galvin, and Greg Gagne. *Operating system concepts*. 10th

Outline

Process Operations

- ▶ An OS must provide mechanisms for,
    - ▶ process creation,
    - ▶ process termination, and others
- ▶ Design considerations and related concepts.

System Calls for Process Operations in UNIX

- ▶ Process creation. fork(), exec(), wait() ...
- ▶ Process termination. exit(), wait(), abort() ...
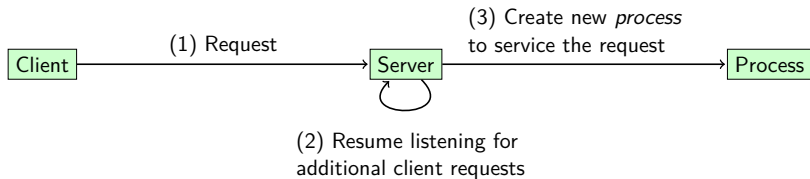
Multiprocess Architecture

Figure: A multiprocess architecture server program

Design Multiprocess Architecture Programs

Design programs of multiprocess architecture on UNIX ...

▶ Observe the example programs and consider rationale behind the design ...

Outline

Examining Process States on Linux Systems

▶ Recall previous discussion ...
  ▶ ps, pstree, top, vmstat, iostat, lsof

, ...

▶ The /proc file system

man proc

References I

Silberschatz, Abraham, Peter B. Galvin, and Greg Gagne. *Operating system concepts*. 10th edition. John Wiley & Sons, 2018.