Operating Systems and Networks

Lecture 04:

Introduction to OS-part 3

Behzad Bordbar

Recap

- ❑Study of I/O devices
- ❑Device Controller
- ❑How OS manages multiple tasks
- ❑System call and their use in user programs
- ❑API and examples of API

Contents

- ❑Different types of system call
- ❑process
- ❑process =/= program
- ❑Stack and heap
- ❑process control block

Different types of system call

- ❑process control
- ❑file manipulation
- ❑device manipulation
- ❑information maintenance
- ❑communications
- ❑protection.

System call: process control

process =/= program

an instance of a running program (two process can be associated to the same program).

- ❑end, abort
- ❑load, execute
- ❑create process, terminate process
- ❑get process attributes, set process attributes
- ❑wait for time
- ❑wait event, signal event
- ❑allocate and free memory

We will discuss some of these.

end and abort

- ❑Stopping a running program:
- ❑normal: end()
- ❑abnormally: abort().
- ❑abnormally: division to zero, file not available:
- ❑a dump of memory is created for
- ❑debugging (code has problem)
- ❑finding cause (unauthorised access)

Exercise: read about stdout and stderr

load(), execute(), createProcess()

- $firefox will result in loading of firefox from another program (terminal)
- Two issues
- what happens when loading program terminates?
- should I keep the current program running? (backgroud program &)

Creating and managing processes

- ❑If we create a process we can control its execution
- ❑ getting attributes
- ❑setting attributes
- ❑maximum allowable execution time
- ❑ terminate process if we find that it is incorrect or is no longer needed
- ❑wait for an event to happen wait_event() and take action signal_event()

Critical section

- this file is being modified by another program!
- Critical section: two or more proccess access the same data… one must access at a time
- operating systems often provide system calls allowing a process to lock shared data. Then, no other process can access the data until the lock is released.
- acquire_lock() and release_lock()

Hard topic: semaphore and monitors

File Management

- ❑create file, delete file
- ❑ open, close
- ❑ read, write, reposition
- ❑ get file attributes, set file attributes

Device management

process may need multiple resources to execute:

Example: access to particular disk

- ❑request device, release device
- ❑read, write, reposition
- ❑get device attributes, set device attributes
- ❑logically attach or detach devices

Information maintenance

- get time or date, set time or date
- get system data, set system data
- get process, file, or device attributes
- set process, file, or device attributes

Communications

- ❑How does two process communicate?
- ❑message-passing model
- ❑shared-memory model

message-passing model

- ❑ Java program (one process) connecting a database (second process) to read/write data (JDBC)
- ❑ Before message, a connection must be opened.
- ❑ Naming: host id and id of the process running on the host
- ❑ Connection created and open()
- ❑ permission for communication to take place with an accept connection() call.
- ❑ processes involved are called client and server
- ❑ read_message() and write_message() system calls.
- ❑ close_connection() sys. call terminates communication.

shared memory

- ❑shared black board model
- ❑processes gain access and read and write to a shared part of memeory
- ❑operating system prevent one process from accessing another process’s memory. For shared memory two or more processes agree to remove this restriction.
- ❑Who ensures two process do not read and write same location simultaneously: the two processes.
- ❑Thread (thread =/= process =/= program)

Communications

No matter shared-memory or message-passing, the following system calls are supported:

- ❑ create, delete communication connection
- ❑ send, receive messages
- ❑ transfer status information
- ❑attach or detach remote devices

protection

- mechanism for controlling access to the resources: file, devices, processes, …
- Mostly designed around access control (remember practical lecture)
- get permission
- set permission
- allow_user
- deny_user

system program vs applications

System programs (utilities)

- ❑convenient environment for program development and execution
- ❑interfaces to system calls

Examples:

- File management.
- Status information
- File modification.
- Programming-language support…

system program vs applications

OS are often supplied with applications : programs that are useful in solving common problems or performing common operations.

Example: database, compiler, game…

process

- ❑A program is a passive entity.
- ❑a file containing a list of instructions stored on disk(foo.exe).
- ❑a process is an active entity, with a program counter specifying the next instruction to execute and a set of associated resources.
- ❑A program becomes a process when an executable file is loaded into memory.
- ❑what processes I am running?

```
$ps –el
```

What are these resources?

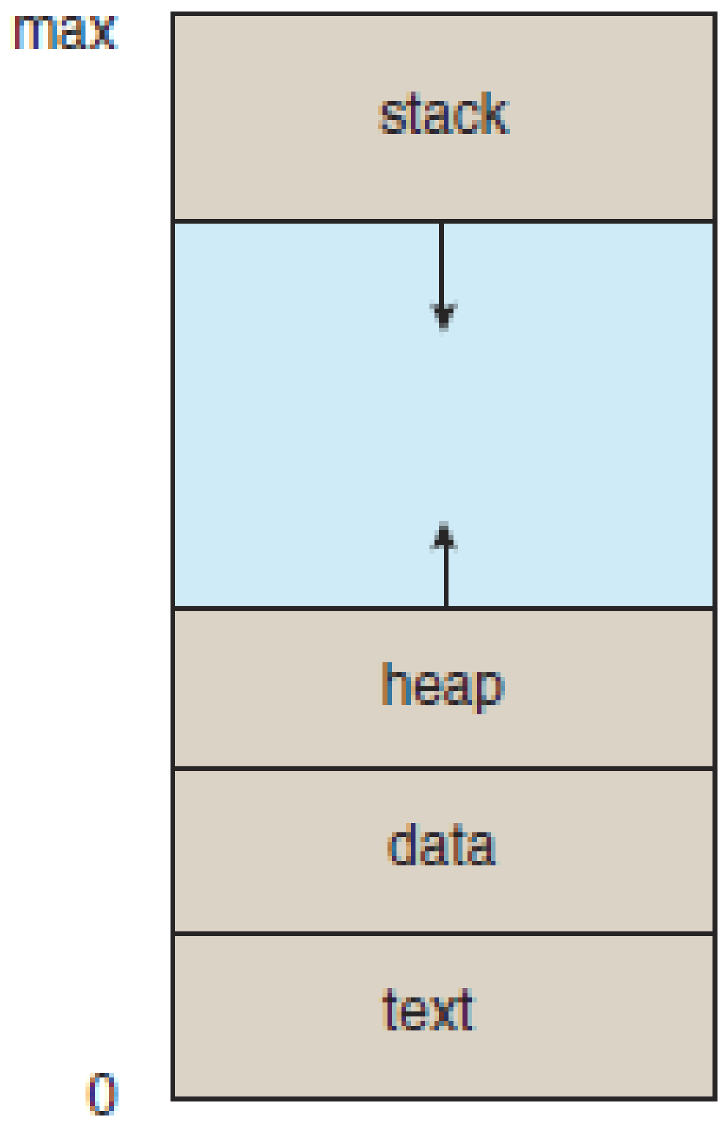
process in memory

A process has the following:

- ❑ text section: the program code, which is sometimes known as the text section.
- ❑ current activity: the value of the program counter + the contents of the processor’s registers.
- ❑ process stack: contains temporary data (such as function parameters, return addresses, and local variables)
- ❑ data section: contains global variables.
- ❑ heap: which is memory that is dynamically allocated during process run time.

heap and stack

- stack frames



picture from dragonbook

stack and heap

- ❑Stack is a contiguous block of memory that is allocated to each process.
- ❑Stack is LIFO (last in first out).
- ❑Stack has stack frame: contains parameter to functions, local vairables and data for recovering previous stack frame.
- ❑When a function is called frame for that function is pushed into stack, when done executing we pop. When we call stack grow, when we execute stack shrinks and we end up with the caller and then frame pops.

stack and heap

- ❑Stack is high performance and is fixed-rate
- ❑in C/C++ and java depending on your code goes to

Java

- ❑new book() puts book on heap frame
- ❑int price put price on stack frame

C/C++

- ❑int familymemeber[10] stack
- ❑int* myArray= new int[10]; puts on heap

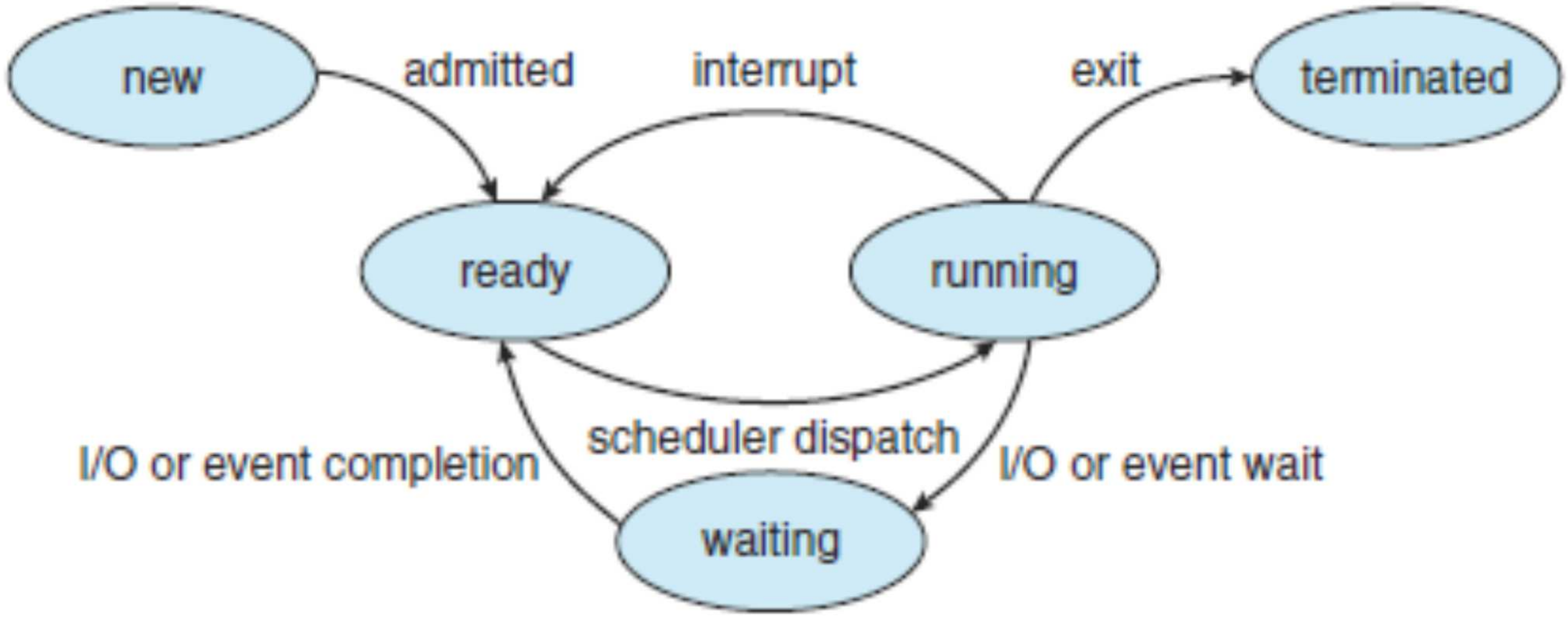
stack and heap

- fast register that we read from memory into them and do the computation and push back to memory. They are very fast.
- Stack register
- ESP (stack pointer) points into top of stack manipulated by pop and push and call (FUNCTION) and RET
- EBP (stack base pointer): this one points to star tof the frame. ESP shows offset from EBP
- EBP fixed and ESP is dynamic
- ESI and EDI source and destination instructions. Where to go? Who called who?

running a process and java

- ❑How do I load a process to memory?
- ❑java is example of a process that is an execution environment for other code.
- ❑JWM is a process that interprets the loaded Java code
- ❑loaded java byte code?
- ❑$javap –c progname

process state



- picture from dragonbook

process control block (PCB)

information required for context switching

❑Process state

❑ Program counter.

❑CPU registers

❑ CPU-scheduling information

❑ Memory-management information

❑Accounting information

❑ I/O status information

summary

- ❑various types of system call
- ❑Process and differences between process and program
- ❑stack and heap
- ❑process state and control block

… now we know what is saved when switching context.