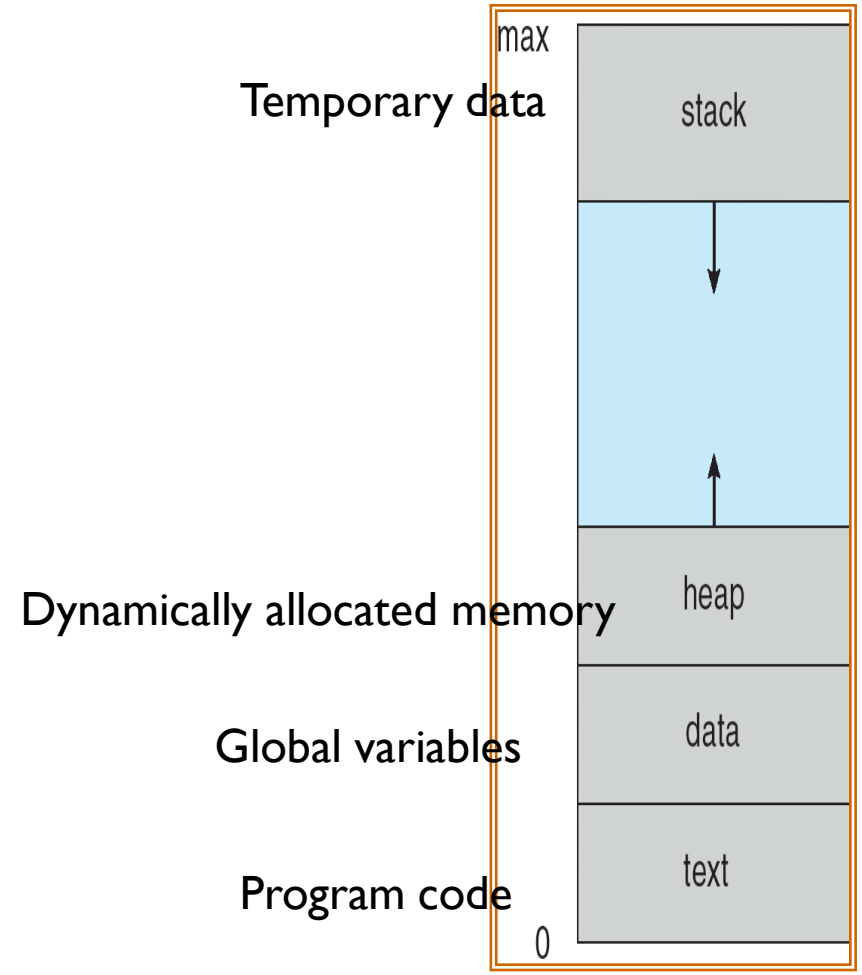
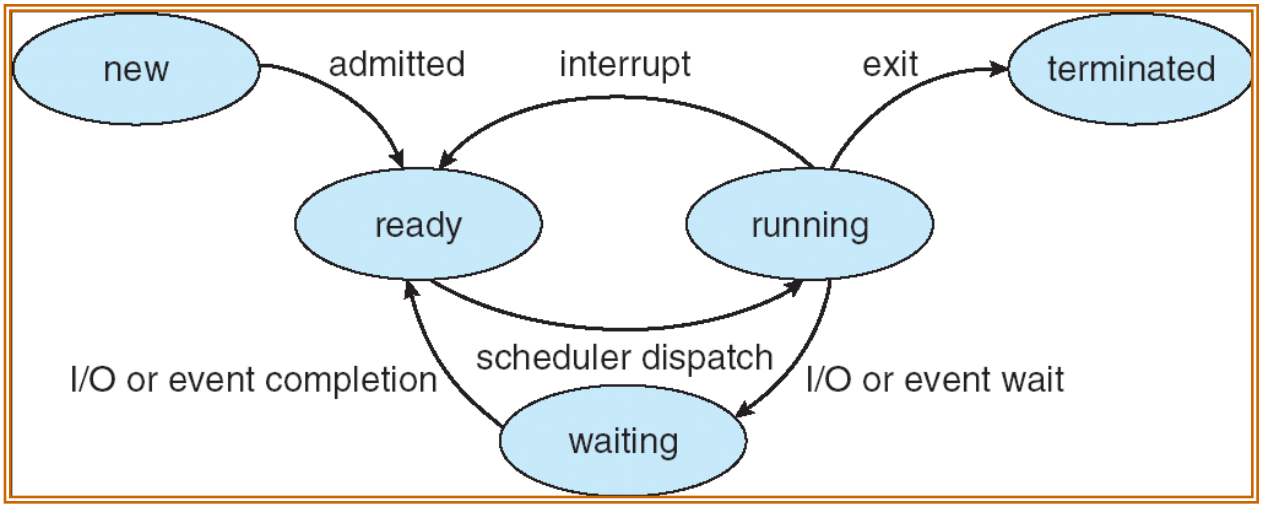
Processes

Process Concept (1)

- A process is a program in execution
  - Active entity
  - Job (batch systems), task (time-shared systems)
  - Program counter + registers
- Program is a passive entity
  - Executable file
- Separate instances of a program are separate processes



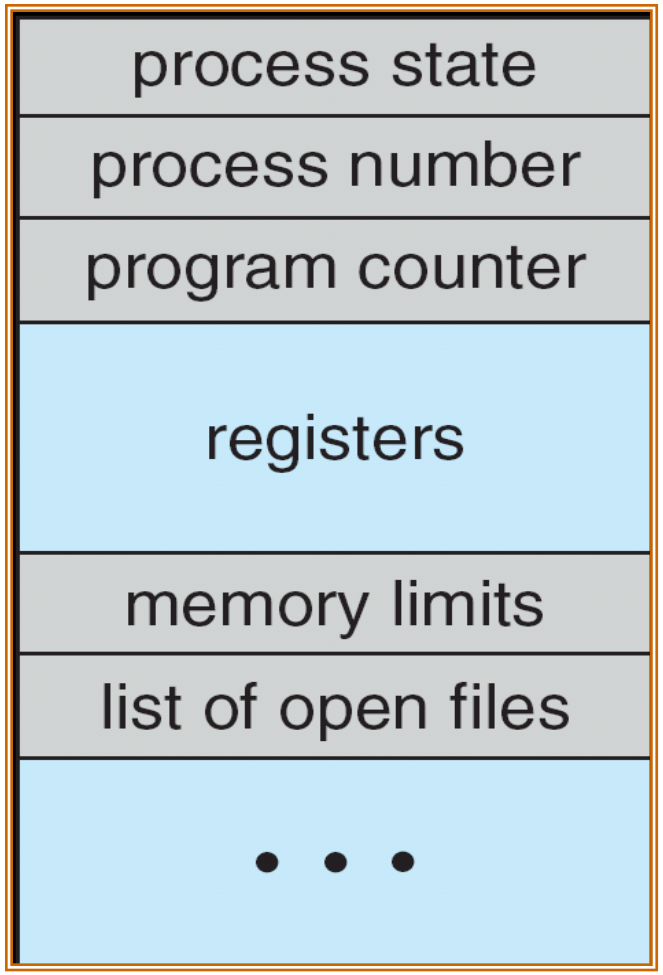
Process Concept (2)



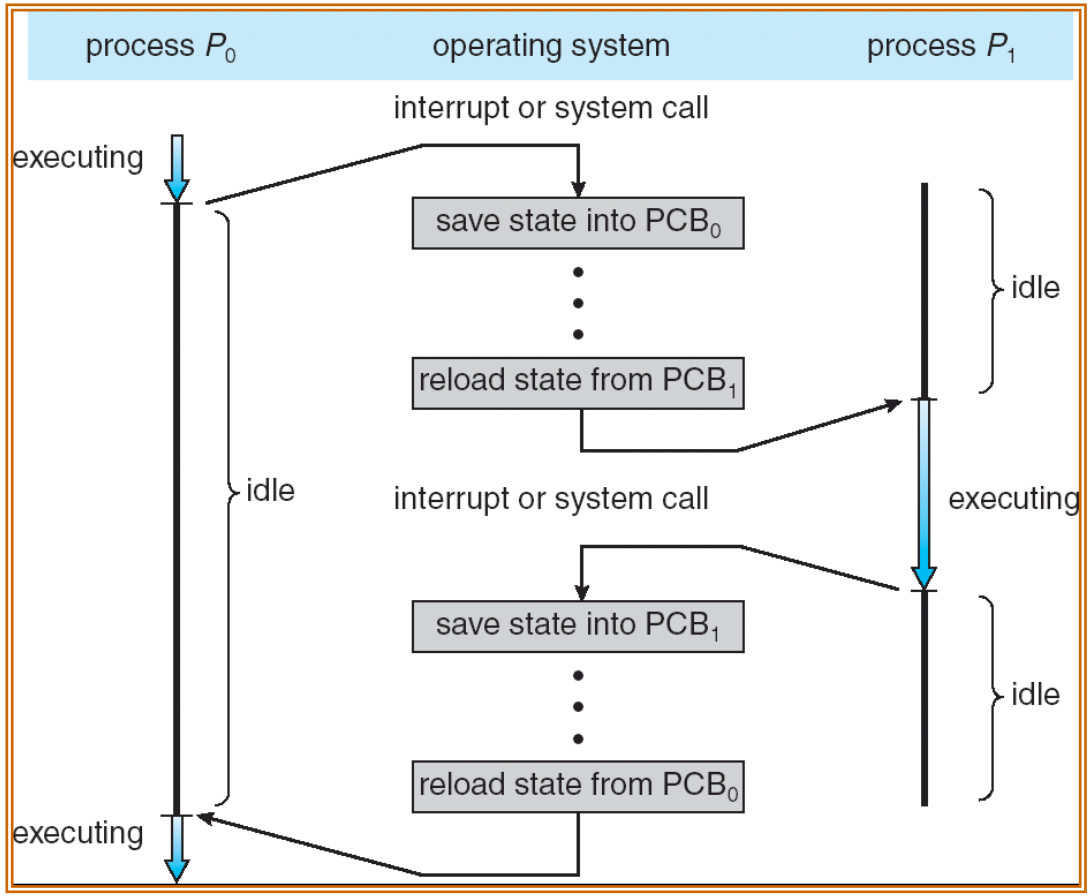
- Only one process running on any processor at any instant
- Many processes may be ready and waiting

Process Concept (3)

- Process control block (PCB)
  - Process state
  - Program counter
  - CPU registers
  - CPU scheduling information
  - Memory-management information
  - Accounting information
  - I/O status information
- Threads of control

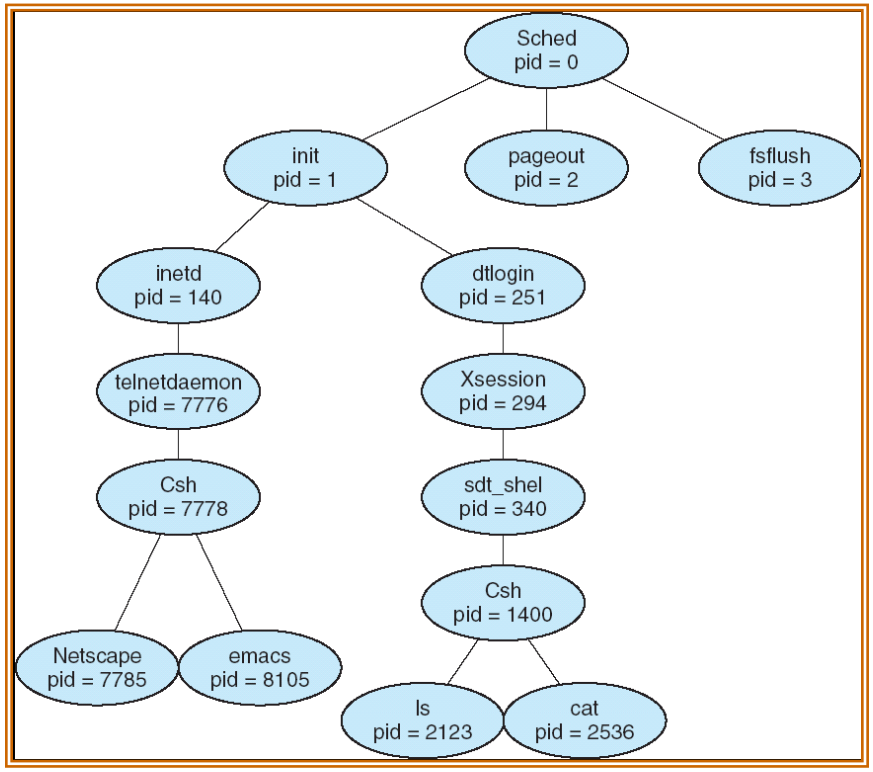


Process Concept (4)



Operations on Processes (1)

- Process creation
  - Tree of processes – parent & children
  - pid – process identifier typically an integer
- Process list
  - ps – el (UNIX)
- Assigning resources
  - New or shared
- Continue execution or wait
- Duplicate or new program



Process Creation in Unix

```
#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>

int main()
{
pid_t pid;

    /* fork a child process */
    pid = fork();

    if (pid < 0) { /* error occurred */
       fprintf(stderr, "Fork Failed");
       exit(-1);
    }
    else if (pid == 0) { /* child process */
       execlp("/bin/ls","ls",NULL);
    }
    else { /* parent process */
       /* parent will wait for the child to complete */
       wait(NULL);
       printf("Child Complete");
       exit(0);
    }
}
```

Process Creation in Windows

```
#include <stdio.h>
#include <windows.h>

int main(VOID)
{
STARTUPINFO si;
PROCESS_INFORMATION pi;

    // allocate memory
    ZeroMemory(&si, sizeof(si));
    si.cb = sizeof(si);
    ZeroMemory(&pi, sizeof(pi));

    // create child process
    if (!CreateProcess(NULL, // use command line
     "C:\\WINDOWS\\system32\\mspaint.exe", // command line
     NULL, // don't inherit process handle
     NULL, // don't inherit thread handle
     FALSE, // disable handle inheritance
     0, // no creation flags
     NULL, // use parent's environment block
     NULL, // use parent's existing directory
     &si,
     &pi))
    {
       fprintf(stderr, "Create Process Failed");
       return -1;
    }
    // parent will wait for the child to complete
    WaitForSingleObject(pi.hProcess, INFINITE);
    printf("Child Complete");

    // close handles
    CloseHandle(pi.hProcess);
    CloseHandle(pi.hThread);
}
```

Operations on Processes (2)

Operations on Processes (3)

- Process creation in Java
  - Java does not support a process model because memory isolation within the JVM is difficult

```
import java.io.*;

public class OSProcess
{
 public static void main(String[] args) throws IOException {
  if (args.length != 1) {
   System.err.println("Usage: java OSProcess <command>");
   System.exit(0);
  }

  // args[0] is the command
  ProcessBuilder pb = new ProcessBuilder(args[0]);
  Process proc = pb.start();

  // obtain the input stream
  InputStream is = proc.getInputStream();
  InputStreamReader isr = new InputStreamReader(is);
  BufferedReader br = new BufferedReader(isr);

  // read what is returned by the command
  String line;
  while ( (line = br.readLine()) != null)
    System.out.println(line);

  br.close();
 }
}
```

Operations on Processes (4)

- Process termination
  - `exit()` - de-allocate resources
  - `TerminateProcess()` – restricted to parent
    - Cascading termination