



SNS COLLEGE OF TECHNOLOGY

Coimbatore-35.

An Autonomous Institution

**Accredited by NBA – AICTE and Accredited by NAAC – UGC with ‘A+’ Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai**



COURSE NAME : 19CST201 – OPERATING SYSTEMS

II YEAR/ IV SEMESTER

UNIT – I OVERVIEW AND PROCESS MANAGEMENT

Topic: Operations on process

Dr.V.Savitha

Assistant Professor

Department of Computer Science and Engineering

Operations on Processes

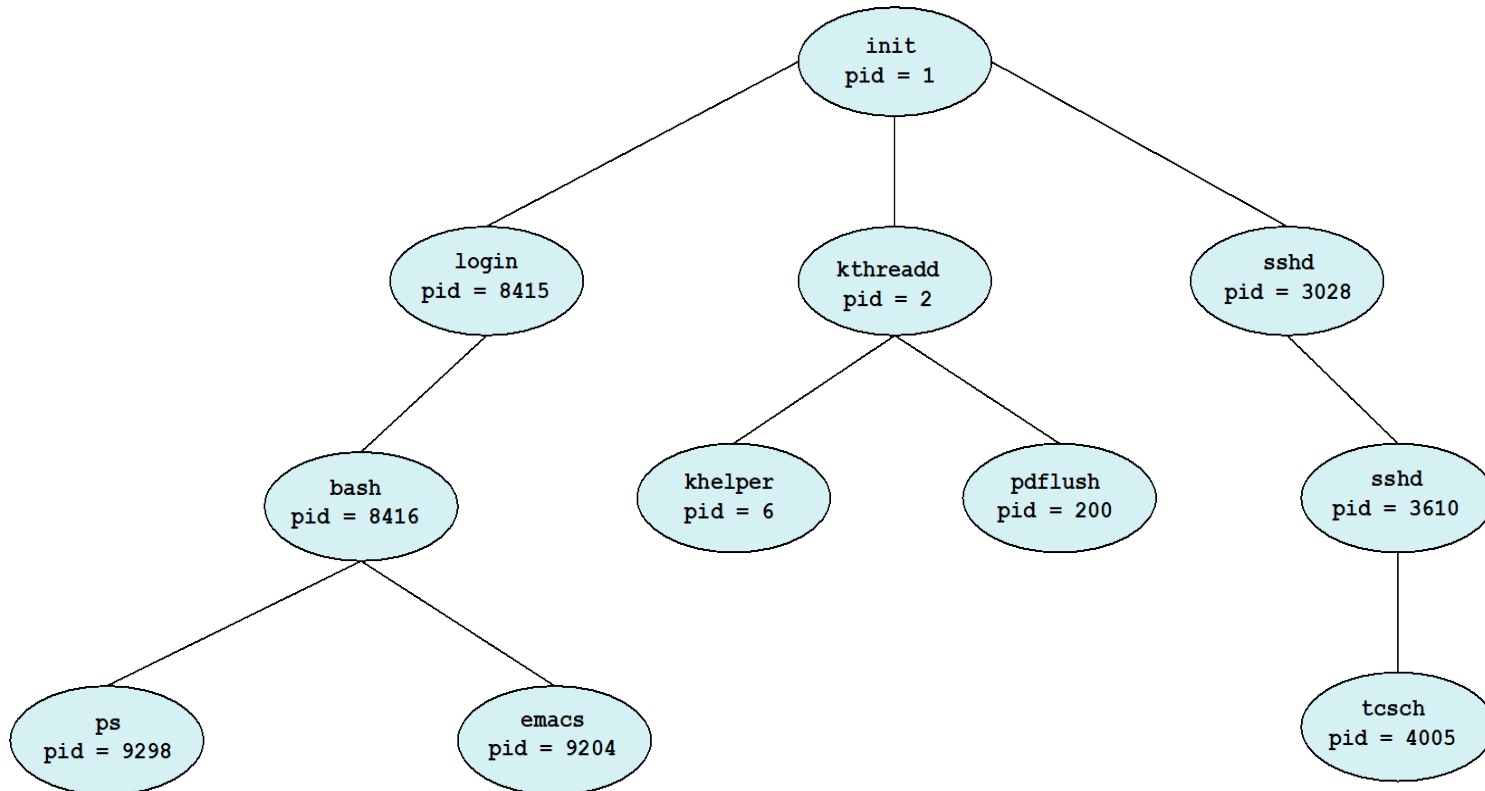


Process Creation

- **Parent** process create **children** processes, which, in turn create other processes, forming a **tree** of processes
- Generally, process identified and managed via a **process identifier (pid)**
- Resource sharing options
 - Parent and children share all resources
 - Children share subset of parent' s resources
 - Parent and child share no resources
- Execution options
 - Parent and children execute concurrently
 - Parent waits until children terminate



A Tree of Processes in Linux

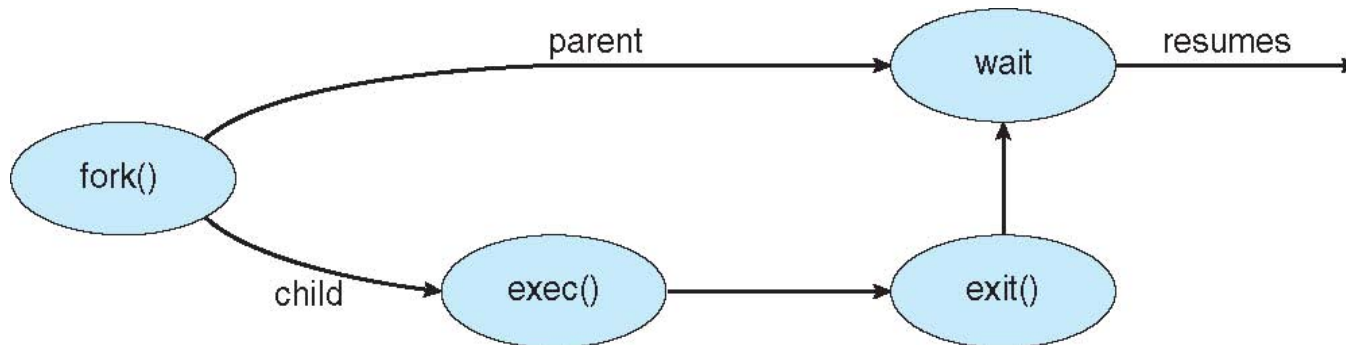




Process Creation (Cont.)



- Address space
 - Child duplicate of parent
 - Child has a program loaded into it
- UNIX examples
 - `fork()` system call creates new process
 - `exec()` system call used after a `fork()` to replace the process' memory space with a new program





C Program Forking Separate Process



```
#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");
        return 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");
    }

    return 0;
}
```



Process Termination



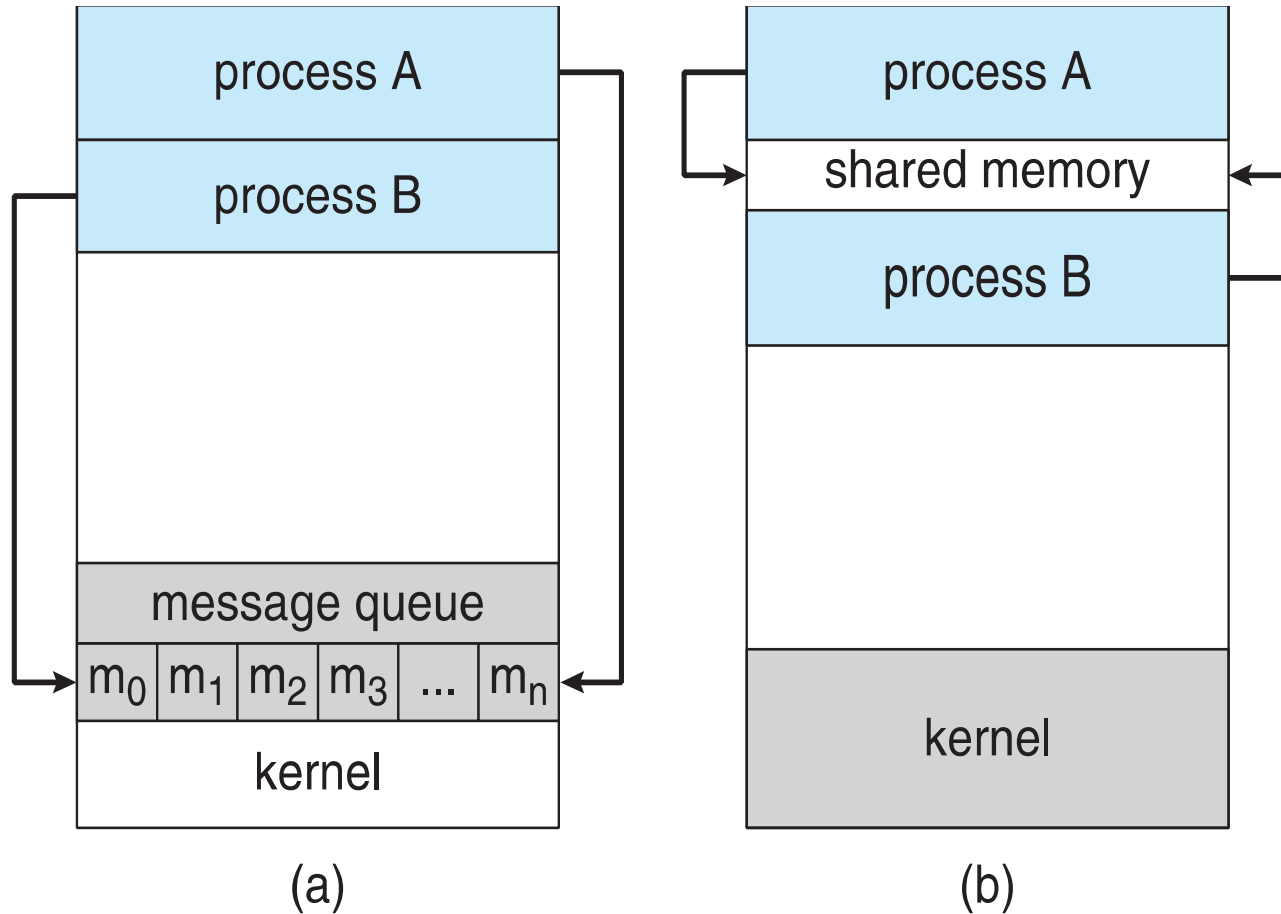
- Process executes last statement and then asks the operating system to delete it using the `exit ()` system call.
 - Returns status data from child to parent (via `wait ()`)
 - Process' resources are deallocated by operating system
- Parent may terminate the execution of children processes using the `abort ()` system call. Some reasons for doing so:
 - Child has exceeded allocated resources
 - Task assigned to child is no longer required
 - The parent is exiting and the operating systems does not allow a child to continue if its parent terminates



Communications Models



(a) Message passing. (b) shared memory.





Cooperating Processes



- **Independent** process cannot affect or be affected by the execution of another process
- **Cooperating** process can affect or be affected by the execution of another process
- Advantages of process cooperation
 - Information sharing
 - Computation speed-up
 - Modularity
 - Convenience



Bounded-Buffer – Shared-Memory Solution



□ Shared data

```
#define BUFFER_SIZE 10
typedef struct {
    . . .
} item;

item buffer[BUFFER_SIZE];
int in = 0;
int out = 0;
```

□ Solution is correct, but can only use BUFFER_SIZE-1 elements



REFERENCES



TEXT BOOKS:

- T1 Silberschatz, Galvin, and Gagne, “Operating System Concepts”, Ninth Edition, Wiley India Pvt Ltd, 2009.)
- T2. Andrew S. Tanenbaum, “Modern Operating Systems”, Fourth Edition, Pearson Education, 2010

REFERENCES:

- R1 Gary Nutt, “Operating Systems”, Third Edition, Pearson Education, 2004.
- R2 Harvey M. Deitel, “Operating Systems”, Third Edition, Pearson Education, 2004.
- R3 Abraham Silberschatz, Peter Baer Galvin and Greg Gagne, “Operating System Concepts”, 9th Edition, John Wiley and Sons Inc., 2012.
- R4. William Stallings, “Operating Systems – Internals and Design Principles”, 7th Edition, Prentice Hall, 2011