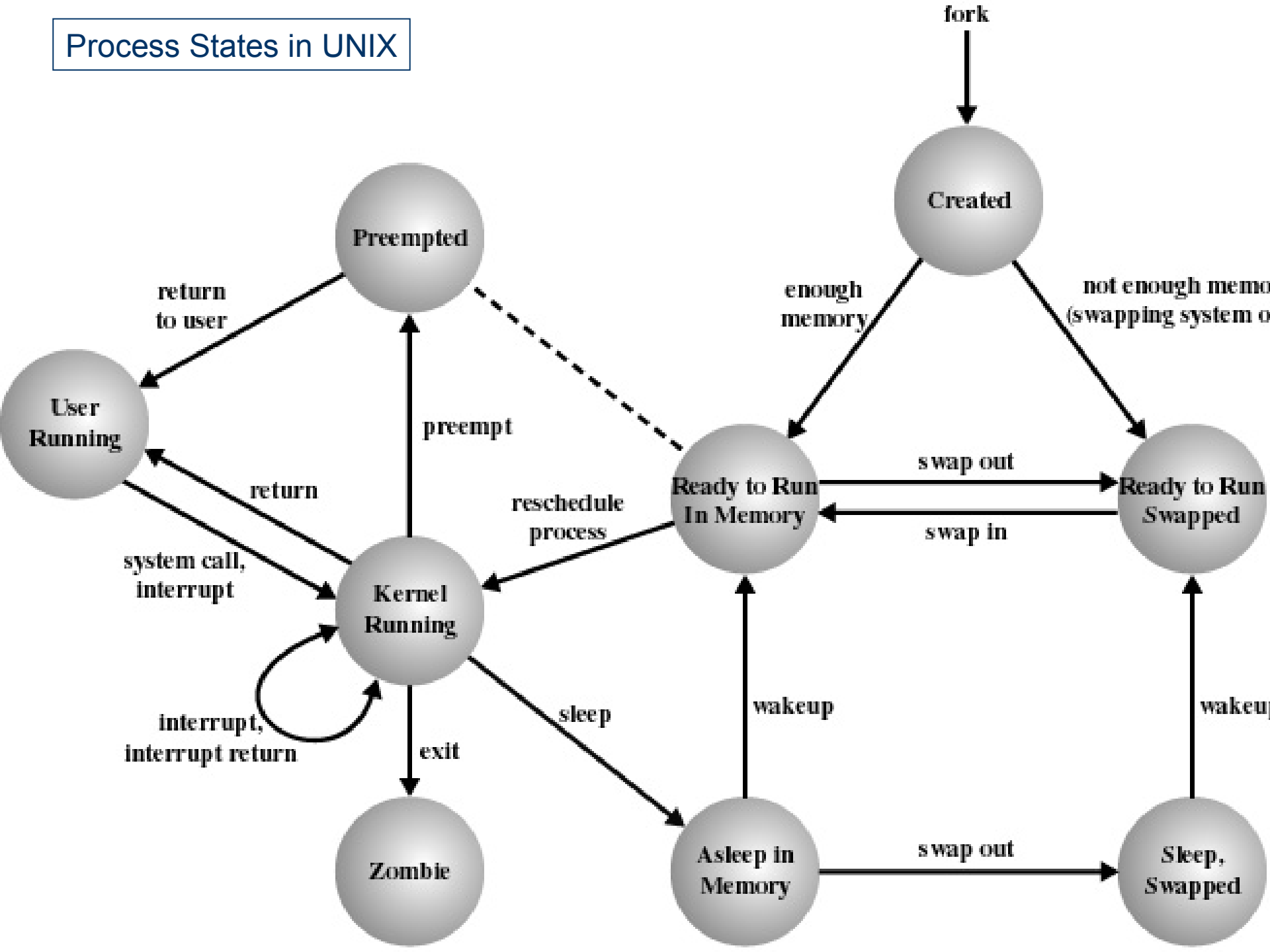
Process States in UNIX



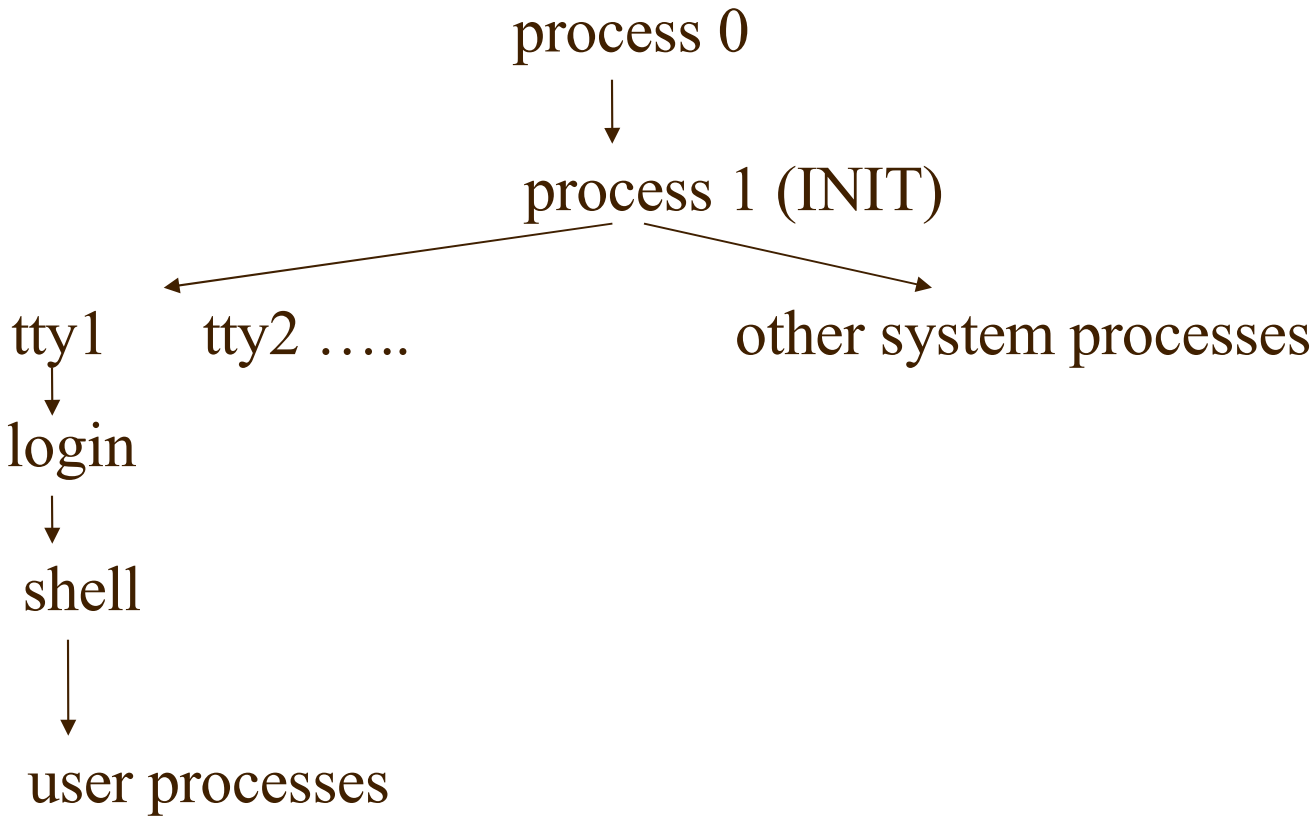
Process Creation in UNIX

- fork system call
  - parent process
  - child process
- syntax: *pid=fork()*
  - both processes have same context
  - returns pid of child to parent process
  - returns 0 to child process
- process no. 0 is the only process not created using *fork*

Process Creation in UNIX

- when fork system call is made:
  - if possible, reserve entry in process table (max no of processes)
  - assign unique pid to child process
  - make a copy of the context of the parent process
  - file access counters modified
  - return child pid to parent process and 0 to child process

Process Hierarchy in UNIX



Exit System Call

- ends execution of process
- syntax: *exit(status)*
  - "*status*" returned to parent process
- all resources returned
- file access counters modified
- process table entry deleted
- when a parent process exits all its children are assigned the init process as a parent (process no 1)

Sample program code - 1

```
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>

int f;

int main (void)
{
  printf("\n Program running: PID=%d \n",
            getpid());
  f=fork();
```

Sample program code - 2

```
if (f==0)  /*child*/
  {
    printf("\nChild process: my pid = %d\n",
            getpid());
    printf("Child process: my parent pid = %d\n",
            getppid());
    sleep(2);
    exit(0):
  }
```

Sample program code - 3

```
else /* parent */
  {
    printf("\nParent process: my pid = %d\n",
            getpid());
    printf("Parent process: my parent pid = %d\n",
            getppid());
    printf("Pranet process: my child's pid = %d\n",
            f);
    sleep(2);
    exit(0):
  }
  return(0);
}
```