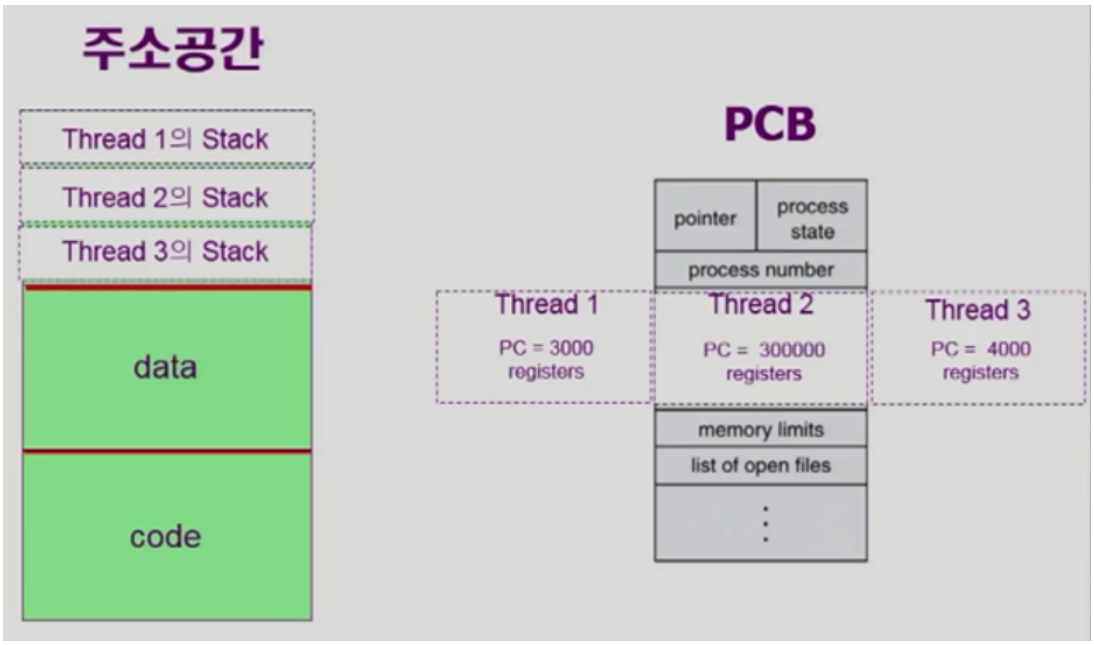
프로세스 관리

생성일	@2022년 3월 26일 오전 11:48
태그	

쓰레드



- 쓰레드는 CPU 실행의 기본 단위
- 구성 : program counter, register set, stack space
- 쓰레드가 동료 쓰레드와 공유하는 부분(=task) : code section, data section, OS resources
- 전통적인 개념의 heavyweight process는 하나의 thread를 가지고 있는 task로 볼 수 있다.
- 다중 쓰레드로 구성된 태스크 구조에서는 하나의 서버 쓰레드가 blocked(waiting) 상태인 동안에도 동일한 태스크 내의 다른 쓰레드가 실행(running)되어 빠른 처리가 가능하다.
- 동일한 일을 수행하는 다중 쓰레드가 협력하여 높은 처리율(throughput)과 성능 향상을 얻을 수 있다.
- 쓰레드를 사용하면 병렬성을 높일 수 있다.

Benefits of Thread

- 응답성이 빠르다
  - multi-threaded web - if one thread is blocked, another thread continues
- 자원 공유 효과
  - n threads can share binary code, data, resource of the process
- 경제성
  - creating & CPU switching 시 overhead 감소
- 멀티 프로세서에서 유리함
  - each thread may be running in parallel on a different processor

Implementation of Threads

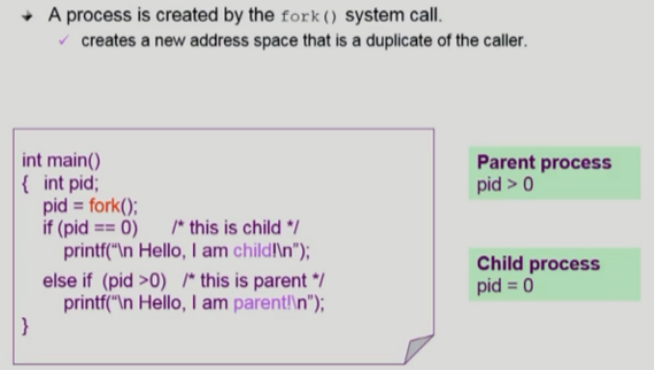
- kernel 기반 Kernel Threads
  - Windows, Solaris, UNIX
- library 기반 User Threads
  - POSIX Pthreads, Mach C-threads, Solaris threads
- real-time Threads

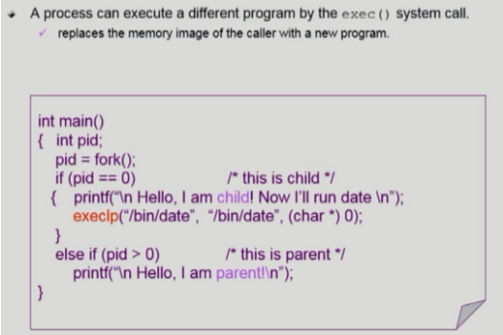
프로세스 생성

- 부모 프로세스가 자식 프로세스를 생성하여 트리 구조가 형성된다.
- 프로세스는 자원을 필요로한다.
  - 운영체제로부터 받는다
  - 부모와 공유한다
- 자원의 공유
  - 부모와 자식이 모든 자원을 공유하는 모델
  - 일부를 공유하는 모델
  - 전혀 공유하지 않는 모델
- 수행(Execution)
  - 부모와 자식은 공존하며 수행되는 모델
  - 자식이 종료(terminate)될 때까지 부모가 기다리는 모델
- 주소 공간(Address space)
  - 자식은 부모의 공간을 복사함 (binary and OS data)
  - 자식은 그 공간에 새로운 프로그램을 올림
- 유닉스의 예
  - fork() 시스템 콜이 새로운 프로세스를 생성
    - **부모를 그대로 복사** (OS data except PID + binary)
    - 주소 공간을 할당
  - fork 다음에 이어지는 exec() 시스템 콜을 통해 **새로운 프로그램을 메모리에 올림**

프로세스 종료

- 프로세스가 마지막 명령을 수행한 후 운영체제에게 이를 알려줌(exit)
  - 자식이 부모에게 output data를 보냄 (via wait)
  - 프로세스의 각종 자원들이 운영체제에게 반납됨
- 부모 프로세스가 자식의 수행을 종료시킴 (abort)
  - 자식이 할당 자원의 한계치를 넘어섬
  - 자식에게 할당된 태스크가 더 이상 필요하지 않음
  - 부모가 종료(exit)하는 경우
    - 운영체제는 부모 프로세스가 종료하는 경우 자식이 더 이상 수행되도록 두지 않고 단계적으로 종료한다



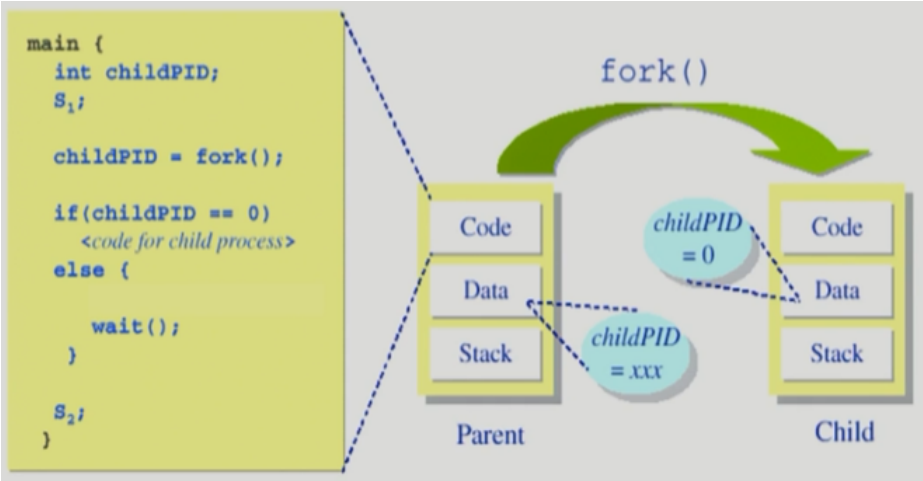


프로세스와 관련한 시스템 콜

- fork() : create a child (copy)
- exec() : overlay new image
- wait() : sleep until child is done
- exit() : frees all the resources, notify parents

wait() 시스템 콜

- 프로세스 A가 wait() 시스템 콜을 호출하면
  - 커널은 child가 종료될 때까지 프로세스 A를 sleep시킨다 (block 상태)
  - Child process가 종료되면 커널은 프로세스 A를 깨운다 (ready 상태)



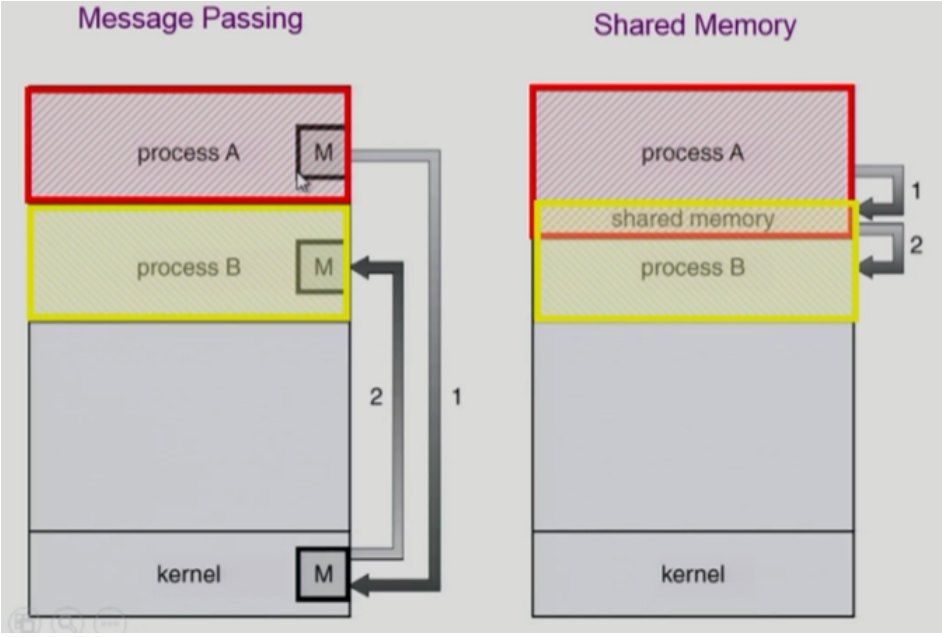
exit() 시스템 콜

- 프로세스의 종료
  - 자발적 종료
    - 마지막 statement 수행 후 exit() 시스템 콜을 통해
    - 프로그램에 명시적으로 넣어주지 않아도 main함수가 리턴되는 위치에 컴파일러가 넣어줌

- 비자발적 종료
  - 부모 프로세스가 자식 프로세스를 강제 종료시킴
    - 자식 프로세스가 한계치를 넘어서는 자원을 요청할 때
    - 자식에게 할당된 태스크가 더 이상 필요하지 않을 때
  - 키보드로 kill, break 등을 친 경우
  - 부모가 종료하는 경우
    - 부모 프로세스가 종료하기 전에 자식들이 먼저 종료

프로세스 간 협력

- 독립적 프로세스 (Independent Process)
  - 프로세스는 각자의 주소 공간을 가지고 수행되므로 원칙적으로 하나의 프로세스는 다른 프로세스의 수행에 영향을 미치지 못함
- 협력 프로세스 (Cooperating Process)
  - 프로세스의 협력 매커니즘을 통해 하나의 프로세스가 다른 프로세스의 수행에 영향을 미칠 수 있음
- 프로세스 간 협력 매커니즘 (IPC: Inter-process Communication)
  - 메세지를 전달하는 방법
    - message passing : 커널을 통해 메세지 전달
  - 주소 공간을 점유하는 방법
    - shared memory : 서로 다른 포르세스 간에도 일부 주소 공간을 공유하게 하는 shared memory 매커니즘이 있음



*** thread는 사실상 하나의 프로세스이므로 프로세스 간 협력으로 보기는 어렵지만 동일한 process를 구성하는 thread들 간에는 주소 공간을 공유하므로 협력이 가능!

Message passing

- 프로세스 사이에 공유 변수(shared variable)를 일체 사용하지 않고 통신하는 시스템
- Direct Communication

    - 통신한려는 프로세스의 이름을 명시적으로 표시
- Indirect Communication
    - mailbox(또는 port)를 통해 메세지를 간접 전달