**OS 중간고사 정리**

Chapter3. Process Concept

Process: 프로그램을 실행하는 주체

**프로세스 수행 상태**

Ready: 실행 될 준비.

Running: 실행 중.

Waiting: 이벤트가 발생하기를 기다리는 상태.

Scheduling: 실행시킬 프로세스를 조건에 따라 선택하는 것.

Dispatch: 선택된 프로세스를 CPU에 올리는 작업. CPU 레지스터들 설정

Ready → running: 스케쥴링을 통해 실행될 프로세스를 골라 CPU에 올린다.

Running → waiting: 이벤트 발생 대기시 상태변경. CPU가 비워지기 때문에 다른 프로세스 실행

Running → ready: 인터럽트 발생하면 수행 중이던 프로세스는 다시 ready상태

Context switch: CPU가 수행하던 프로세스를 다른 프로세스로 교체하는 것. (=CPU Switch)

오버헤드 발생! 그러나 프로세스를 나눠서 수행하면 응답시간이 짧아지기 때문이다. (time sharing)

시험 출제

**프로그램 실행 예시**

HWP MP3 idle

Context Switch

Re Re Re 전체 대기 상태

Re R Re 스케쥴링에 따라 MP3 실행

R W Re 디스크에서 파일을 불러오기 위해 W 변경, 스케쥴링에 따라 HWP 실행

W W R 키보드 입력 대기 W 변경, 수행할 프로세스가 없으므로 idle 실행

W Re Re 디스크에서 파일 읽기를 완료하여 인터럽트 발생, Re 변경.

W R Re 스케쥴링에 따라 MP3 실행 (디코딩)

Re Re Re I/O 인터럽트 발생 (키보드 입력)

R Re Re 스케쥴링에 따라 HWP 실행

W R Re 키보드 입력 대기 W 변경, 스케쥴링에 따라 MP3 실행

︙ ︙ ︙

\*idle: 무한루프 프로세스

**프로세스 생성 및 종료**

Fork(): 자신과 똑같은 프로세스를 만들어 냄. Parent와 협력하면서 동작할 때 좋음!

Exec(): 프로그램을 실행시키는 함수.

Fork의 트리 맨 위에는 Root는 운영체제가 만들었음. (UNIX에서, windows는 트리 없음)

Fork를 수행하는 순간 두 개의 프로세스가 생겨 각각 실행된다.

Fork의 Return value = 0: child

Return value > 0: parent

CreateProcess: windows의 프로세스 생성 함수. (fork+exec)

Exit: 프로그램 종료

Abort: 프로그램 비정상 종료

**프로세스간의 커뮤니케이션**

|  |  |  |
| --- | --- | --- |
|  | Message Passing | Shared Memory |
| 설명 | 커널이 중재, 커널에 데이터를 저장하고 데이터를 읽어간다. | 공유 메모리를 두고 같이 사용한다. |
| 장점 | 커널이 데이터의 존재를 알고 알려주기 때문에 동기화가 필요 없다. | 성능이 좋다. |
| 단점 | 성능이 떨어진다. | 동기화가 필요하다.  문제 1: 동시에 메모리에 접근할 경우  문제 2: 메모리를 저장하기 전에 읽어가려고 할 경우 |

Direct Communication: 보낼 때 받는 Process를 지정.

Indirect Communication: 받는 프로세스를 지정하는 것이 아니라 저장해두는 메모리 장소를 지정.

대부분의 OS가 후자를 채택!

**동기화**

Blocking (synchronous): 메시지가 있을 때까지 기다림. (현재 default)

Non-blocking (asynchronous): 다른 프로그램을 실행하고 있음.

RPC/RMI : 다른 PC에 있는 함수를 호출