

# 프로세스(Process)

발표자: 김건희

# 목차

1. 프로세스란 무엇일까요?
2. 프로그램, 프로세스, 스레드 비교
3. PCB
4. Linux의 프로세스/스레드 생성
5. 좀비 & 고아 프로세스
6. 데몬과 init
7. Init 프로세스와 프로세스 트리

# 1. 프로세스란 무엇일까요?

프로그램이 메모리에 올라와 CPU에 의해 실행되고 있는 상태를 의미합니다. 정적인 프로그램 파일이 동적으로 실행되며 시스템 자원을 할당받은 작업 단위입니다.



## 변화

정적 상태(프로그램) → 동적 실행(프로세스)  
메모리 적재 + CPU 할당 = 프로세스

# 프로그램, 프로세스, 스레드 비교

## 3가지 핵심 개념의 차이점

운영체제에서 프로그램, 프로세스, 스레드는 밀접하게 관련되어 있지만 각각 다른 역할과 특성을 가집니다. 이들의 차이점을 이해하면 컴퓨터 시스템의 실행 구조를 더 명확히 파악할 수 있습니다.

### 프로그램

실행 가능한 정적 파일  
디스크에 저장된  
코드 덩어리

### 프로세스

실행 중인 프로그램  
독립적인 메모리 공간  
자원 할당의 단위

### 스레드

프로세스 내 실행 흐름  
자원 공유  
경량 프로세스







# 3. PCB

운영체제가 프로세스를 식별하고 관리하기 위한 데이터 구조로 각 프로세스마다 하나씩 생성됩니다.

“

프로세스가 실행 중일 때는 PCB가 메모리에 유지되고, 프로세스가 종료되면 PCB도 제거됩니다.

## PCB에 포함된 주요 정보:

-  프로세스 식별자 (**PID**) - 각 프로세스의 고유 번호
-  프로세스 상태 - 생성, 준비, 실행, 대기, 종료 등 프로
-  그램 카운터 - 다음에 실행할 명령어 주소 레지스터
-  정보 - CPU 레지스터 상태 저장
-  메모리 관리 정보 - 메모리 위치, 페이지 테이블 등
-  입출력 상태 정보 - 프로세스에 할당된 입출력 자원

? 질문: 그렇다면 스레드도 **PCB**를 가질까요?

# 3-1. 스레드와 PCB의 관계

스레드는 PCB를 직접 가지지 않고, **TCB(Thread Control Block)**라는 더 작은 제어 블록을 통해 관리됩니다. 스레드는 프로세스의 자원을 공유하기 때문에, 프로세스 전체 정보는 PCB에만 저장하고 TCB에는 실행에 필요한 최소한의 정보만 저장합니다.

## PCB (Process Control Block)

- PID (프로세스 식별자)
- 프로세스 상태
- 프로그램 카운터
- CPU 레지스터 정보
- CPU 스케줄링 정보
- 메모리 관리 정보
- 입출력 상태 정보
- 계정 정보

프로세스 관리 정보

## TCB (Thread Control Block)

- 스레드 식별자
- 스레드 상태
- 프로그램 카운터
- 레지스터 집합
- 스택 포인터
- 우선 순위

스레드 실행 정보

# 4. Linux에서의 생성 방법

## 프로세스 vs 스레드 생성:

리눅스에서 프로세스와 스레드는 서로 다른 시스템 콜을 사용해 생성합니다. 프로세스는 별도의 메모리 공간을 할당받지만, 스레드는 자원을 공유합니다.

- 프로세스 생성은 부모의 복제를 통해 이루어짐
- 스레드는 프로세스 내에서 공유 자원을 활용 내부적
- 리눅스는 으로 모두 **clone()** 시스템 콜 사용

01

리눅스의 프로세스와 스레드 생성 원리

02

프로세스: `fork()`

부모 프로세스를 복제하여 새로운 주소  
공간 생성

03

스레드: `pthread_create()`

동일 주소 공간 내 새로운 실행 흐름  
생성

# 좀비 & 고아 프로세스

## 특수한 프로세스 상태와 해결 방법

프로세스 종료 과정에서 발생하는 특수 상황들과 운영체제가 이를 처리하는 방식을 알아봅니다.

좀비 프로세스: 자식 프로세스가 종료되었지만 부모 프로세스가 `wait()` 호출을 하지 않아 상태 정보가 남아있는 상태

고아 프로세스: 부모 프로세스가 자식보다 먼저 종료되어 부모가 없는 상태가 된 프로세스

해결 방법: `init` 프로세스(PID 1)가 고아 프로세스의 새 부모가 되어 종료 상태를 회수함



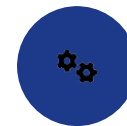
## 6. 데몬 프로세스 (Daemon Process)



### 백그라운드 서비스

#### 데몬이란?

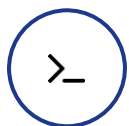
사용자와 직접 상호작용 없이 백그라운드에서 동작하는 프로세스입니다. 주로 시스템 서비스를 제공하며 시스템 부팅시 자동으로 실행되어 종료될 때까지 계속 실행됩니다.



### 무한대기 모드

#### 실행 방식

독립적으로 실행되는 Stand-alone 방식과 슈퍼데몬 (xinetd)이 요청시 실행하는 방식으로 나뉩니다. 터미널과 연결되지 않으며(TTY 없음), PPID가 1인 경우가 많습니다.



### 대표 예시

httpd (웹서버)

sshd (원격접속)

systemd (시스템관리)

ftpd (파일전송)

syslogd (로깅)

mysqld (데이터베이스)

# init 프로세스와 프로세스 트리

## 리눅스 프로세스 관리의 핵심

리눅스에서 모든 프로세스는 계층적 트리 구조를 형성하며, init 프로세스(PID 1)는 그 뿌리가 됩니다. systemd가 현대 리눅스에서 init의 역할을 대체하고 있습니다.

시스템 부팅 시 최초  
프로세스

모든 프로세스의 최상  
위 조상

고아 프로세스 자동 입  
양 및 관리

