운영 체제

CH 0.

DAY-1

* 기본적인 동작

1. 파일의 내용을 저장할 공간을 할당 받고 저장. (= Disk Block, 일반적으론 4kb 단위)
2. Disk에 inode라는 파일의 정보(만들어진 시간, 사람, 접근 제어 정보 등)을 담고 있는 공간을 할당 받음.
3. Inode와 파일을 연결함
4. 컴파일러를 통해 파일을 수행 가능한 바이너리 파일(0,1로 이루어진 파일)로 만듦.
5. 바이너리 파일을 실행할 때에는 task라는 객체를 만듦.
6. task라는 객체를 만들기 위해 바이너리 파일을 우선 RAM에 load함.
7. Task는 RAM의 일부 공간(page frame)을 할당 받고 관리.
8. 기존에 존재하던 task들과 경쟁하며 CPU를 할당받기 위해 노력함.
9. 운영체제는 이런 task들에게 CPU를 할당해줌. (대표적으로 Round-Robin[[1]](#footnote-1) 방식이 유명)

* 임베디드 부팅을 위해선 3개의 파일이 필요하다.

1. Boot loader: 장치들을 초기화해주고 커널, 파일 시스템을 램에 복사해주는 기능
2. Kernel: 프로그램에 대한 자원 할당을 담당한다. 인터럽트 처리기와 스케줄러 등을 포함한다. 주기억장치에 저장된다.

zImage와 uImage는 리눅스 커널 중 하나다. uImage는 zImage의 특정 명령어를 압축한 것이다.

1. 하나의 압축 파일로, 기본적인 명령어, 라이브러리 파일을 압축하여 파일 시스템으로 만든다.

* Daemon: 한번만 실행해 놓으면 지속적으로 동작하는 프로그램 또는 명령어
* Shell: 이용자와 시스템 사이의 대화를 가능하게 해주는 명령 해석기. 이용자가 입력한 문장을 읽어, 그 문장이 요청하는 시스템 기능을 수행하게 해준다. Kernel과 달리 보조 기억 장치에도 저장이 될 수 있다.
* Idle 상태: 하드웨어 장치에서 다음 수행 명령을 기다리는 동안 별다른 작업을 수행하지 않고 기다리는 상태
* Super Block: 유닉스 시스템에서 파일 시스템의 상태를 설명하는 블록
* metadata: 다른 데이터를 설명해주는 데이터. Contents의 위치와 내용, 작성자 정보, 이용 조건, 내력 등이 담겨있다. metadata는 보통 데이터를 표현하기 위한 목적과 데이터를 더 빠르게 찾기 위한 목적이 있다.

CH1.

* Microkernel[[2]](#footnote-2): 운영체제의 기본적인 기능을 제공하는 kernel만 남기고 소형화한 것.
* Linux는 monolithic structure[[3]](#footnote-3)를 기반으로 설계되었다. (실제로 kernel 자체는 monolithic이지만 파일 시스템, 디바이스 드라이버, 스케줄링 등 여러 부분에서 모듈을 지원하기에 하이브리드라고 볼 수 있다.)

CH2.

Day-2

* OS = resource manager (system call을 통해 task가 resource를 활용할 수 있게 해준다.)

1. Physical resource

CPU, Memory, Disk, terminal, network 등 시스템 구성요소와 주변 장치

1. Abstract resource (matched with physical)

물리적 자원을 관리하기 위한 추상화 객체들

Ex) task, memory segment, page, protocol, packet

1. Abstract resource (Not matched with physical)

Security, ID access control

* Kernel

1. task manager

task의 생성, 실행, 상태 전이, 스케줄링, 시그널 처리, 프로세스 간 통신 지원

1. memory manager

Segment[[4]](#footnote-4)와 page[[5]](#footnote-5) 관리

1. filesystem

파일 생성, 접근 제어, inode 생성, 디렉터리 관리, super block 관리

1. network manager

네트워크 장치를 socket으로 제공, TCP/IP 같은 통신 프로토콜

1. device driver manager.

주변 장치를 일관되게 접근하도록 해줌

1. 일정시간 동안 한 task가 CPU를 사용하고 시간이 지나면 다음 task가 사용하는 방식 [↑](#footnote-ref-1)
2. Windows NT, Mac OS에서 사용 [↑](#footnote-ref-2)
3. 모듈화와는 다르게 독립성이 없는 소프트웨어 블록들의 집합으로 이루어진 소프트웨어.

   Monolithic는 최적화에 좋고, 속도가 빠르며, 경로를 단축시킬 수 있지만 유지보수가 힘들다. [↑](#footnote-ref-3)
4. 주 기억장치에 load되는 프로그램 분할의 기본 단위 [↑](#footnote-ref-4)
5. 프로그램을 한번에 처리할 수 있는 단위. Page 단위로 주 기억 장치에 로드하고 언로드한다. [↑](#footnote-ref-5)