TI DSP,MCU 및 Xilinux Zynq FPGA

프로그래밍 전문가 과정

이름	문지희
학생 이메일	mjh8127@naver.com
날짜	2018/4/6
수업일수	32 일차
담당강사	Innova Lee(이상훈)
강사 이메일	gcccompil3r@gmail.com

목차

Chapter 1 리눅스 소개

(1) 리눅스의 탄생
리누스 토발즈라는 사람이 리차드 스톨만과 GNU 단체와 함께 만든 운영체제이다.
자유 소프트웨어 재단은 컴퓨터 전반의 모든 분야에 공개(free) 소프트웨어를 개발하고 사용하게 함으로써 상용 소프트웨어 프로그램에 대한 복사, 배포, 이해, 수정에 대한 모든 제한을 없애기 위해 노력하는 단체이다.
프리 소프트웨어로는 돈을 벌 수 없으며 GPL 라이센스는 코드는 공개하지만 돈을 벌 수 있다. 여러 라이센스들이 있고 잘 선택해야한다.
리눅스는 원하는 사람은 누구나 그 소스를 수정하여 성능을 향상시킬 수 있으며 내용도 공개해야한다라는 정신이 있다.

(2) 유닉스와 리눅스
생명을 제공하는 태스크와 장소를 제공하는 파일이라는 두 가지 객체로 모든 것을 지원한다 → 모든것은 파일이라는 리눅스의 핵심 철학
리눅스는 모널리식 구조로 만들어져있다. 모놀리식 커널은 탈부착이 불가능한데 디바이스 드라이버에서 만큼은 마이크로 커널로 만들게 되었다. 그래서 USB 를 꽂았다 뺐을 때 컴퓨터가 재부팅이 안되는 이유이다.

(4) 리눅스의 장점
- 사용자 임의대로 재구성이 가능하다. 커스터 마이징이 가능
- 열악한 환경에서도 HW 자원을 적절히 활용하여 동작하기에 성능이 좋지 않은 컴퓨터에서도 리눅스가 깔린다. 모놀리식 방식이기 때문이다.
- 커널의 크기가 작다. 윈도우보단 리눅스가 작음
- 완벽한 멀티유저, 멀티태스킹 시스템
- 네트워크 장비나 기상 기지국에서 사용될 정도로 뛰어난 안정성을 가지고 있다.
- 빠른 업그레이드
- 강력한 네트워크 지원
- 스톨만의 GNU 덕분에 풍부한 소프트웨어를 가지고있다.

사용자를 위한 여러가지 공개 문서들

프로그래머들이 가져야 할 올바른 자세 : 운영체제의 소스 자체가 오픈 되어 있으니 안된다면 내가 되게하면 된다는 마인드.

Chapter 2 리눅스 커널 구조

(1)리눅스 커널 구조
커널 : 운영체제의 핵심을 이루는 부분으로, CPU 나 메모리, 기타 디바이스 등의 시스템 리소스를 관리하고 사용자 프로그램이 이를 사용할 수 있도록 한다. →시스템 콜. 사용자가 커널과 소통할 수 있는 방법은 시스템 콜 밖에 없다.
CPU 를 관리 한다는 건 task 를 관리한다는 의미이고 메모리는 페이지를, 디바이스는 드라이버를 관리한다는 의미이다.

운영체제 : 자원관리자

운영체제가 관리해야 할 자원은 물리적인 자원과 추상적인 자원으로 구분된다.
물리적인 자원 - CPU, 메모리, 디스크, 터미널, 네트워크 등 시스템을 구성하고 있는 요소들과 주변 장치 등
추상적인 자원 - CPU 를 추상화 시킨 task(프로세스), 메모리를 추상화 시킨 세그먼트와 페이지, 디스크를 추상화 시킨 파일, 네트워크를 추상화 시킨 통신 프로토콜, 패킷, 물리적인 자원에 대응되지 않고 추상적인 객체만으로 존재하는 보안과 접근제어도 있다.

태스크 관리자 : 태스크의 생성(fork, pthread_create), 실행(exec), 상태전이(wait queue 인지,run queue 인지), 스케줄링, 시그널 처리, 프로세스 간 통신(IPC) 등의 서비스 제공
메모리 관리자 : 물리메모리 관리(DRAM 관리), 가상메모리 관리(stack, data, heap, text 관리), 세그멘테이션, 페이징, 페이지 부재결함(segmentation fault) 처리 등의 서비스 제공

파일 시스템 : 파일의 생성(open-O_CREAT), 접근제어, inode 관리, 디렉터리 관리, 수퍼 블록 관리 등의 서비스 제공
네트워크 관리자 : 소켓 인터페이스, TCP/IP 같은 통신 프로토콜 서비스 제공
디바이스 드라이버 : 디스크나 터미널, CD, 네트워크 카드 등 주변장치를 구동하는 드라이버들로 구성

-리눅스 커널의 소스트리 구조

kernel 디렉토리

태스크의 생성과 소멸, 프로그램의 실행, 스케줄링, 시그널 처리 등의 기능이 이 디렉토리에 구현되어 있다.

arch 디렉토리

리눅스 커널 기능 중 하드웨어 종속적인 부분들이 구현된 디렉터리이다. architecture 의 줄임말인 arch 이다. 이 디렉터리는 CPU 의 타입에 따라 하위 디렉터리로 다시 구분된다.

```
xeno@xeno-NH:~/kernel/linux-4.4$ ls arch
alpha   avr32      frv      Kconfig  microblaze  openrisc  score  um
arc     blackfin   h8300    m32r     mips        parisc    sh     unicore32
arm     c6x        hexagon  m68k     mn10300     powerpc   sparc  x86
arm64   cris       ia64     metag    nios2       s390      tile   xtensa
```

fs 디렉토리

리눅스에서 지원하는 다양한 파일시스템과 open(), read(), write()등의 시스템 호출이 구현된 디렉토리이다.
가상파일시스템도 이 디렉토리에 존재한다.

mm 디렉토리

메모리 관리자가 구현된 디렉토리이다. 물리메모리관리, 가상 메모리 관리, 태스크마다 할당되는 메모리 객체 관리 등의 기능이 구현되어 있다.

driver 디렉토리
리눅스에서 지원하는 디바이스 드라이버가 구현된 디렉터리이다. 디바이스 드라이버란 디스크, 터미널, 네트워크 카드 등 주변장치를 추상화 시키고 관리하는 커널 구성요소이다. 디바이스 드라이버를 3가지로 구분하는데 파일시스템을 통해 접근되는 블록 디바이스 드라이버, 사용자 수준 응용 프로그램이 장치파일을 통해 직접 접근하는 문자 디바이스 드라이버, TCP/IP 를 통해 접근되는 네트워크 디바이스 드라이버이다.

net 디렉토리
리눅스가 지원하는 통신 프로토콜이 구현된 디렉터리이다. 통신 프로토콜의 추상화 계층이며 사용자 인터페이스를 제공하는 소켓 역시 이 디렉터리에 구현되어 있다.

ipc 디렉토리
리눅스 커널이 지원하는 프로세스간 통신 기능이 구현된 디렉토리이다.

init 디렉토리
커널 초기화 부분, 커널의 메인 시작함수가 구현된 디렉토리이다. 이 함수가 커널 전역적인 초기화를 수행한다.

include 디렉토리
리눅스 커널이 사용하는 헤더 파일들이 구현된 디렉터리이다. 헤더 파일 중에서 하드웨어 독립적인 부분은 include/linux 하위 디렉토리에 정의되어 있고 하드웨어 종속적인 부분은 include/asm-$(ARCH)디렉토리에 정의되어 있다.

others 디렉토리
리눅스 커널의 주요 기능이 구현된 디렉토리 외에도 리눅스 커널 및 명령어들이 자세한 문서 파일들이 존재하는 Documentation 디렉토리, 커널 라이브 함수들이 구현된 lib 디렉토리, 커널 구성 및 컴파일 시 이용되는 스크립트들이 존재하는 scripts 디렉토리 등이 존재한다.

chapter 3 태스크 관리

(1) 프로세스와 쓰레드 그리고 태스크

task_struct 내부를 살펴보게 되면 pid 와 tgid 가 선언 되어있는 것을 볼 수 있다. pid 는 프로세스 아이디이고, tgid 는 쓰레드 그룹 아이디이다. 프로세스와 쓰레드의 차이점은 pid 와 tgid 가 같으면 프로세스이고 pid 와 tgid 가 다르면 쓰레드이다. 쓰레드 리더일 때 프로세스이므로 프로세스도 쓰레드라고 볼 수 있다. fork 를 통하여 프로세스를 만든다는 의미는 pid 와 tgid 가 같은 쓰레드 리더를 만든다는 의미이다. 반면에 pthread_create 로 쓰레드를 만들게 되면 tgid 는 같지만 pid 가 다른 쓰레드가 생성이 된다.

ELF - 리눅스의 실행파일 포맷

DWARF - 리눅스의 디버깅 파일 포맷

프로세스는 동작중인 프로그램이며 커널로부터 할당받은 자신만의 자원을 가지고 CPU 가 기계어 명령들을 실행함에 따라 끈임없이 변화하는 동적인 존재이다.

스케쥴링은 커널이 시스템에 존재하는 여러 개의 프로세스 중 CPU 라는 자원을 어느 프로세스에게 할당해 줄 것인가를 결정하는 작업을 스케줄링이라고 부른다.

(2) 사용자 입장에서 프로세스 구조

사용자 프로세스가 수행되기 위해서는 여러 가지 자원들을 커널로부터 할당 받아야 한다. 프로세스의 구조를 살펴보면 유저 영역과 커널 영역으로 분리되어 있다. 사용자 프로그램 중에서 명령, 함수 등으로 구성되는 텍스트 영역은 프로세스 주소 공간 중 가장 하위공간을 차지한다. 텍스트 영역 다음에는 전역변수 등을 담아 놓는 데이터 영역이 차지한다. 그 위로는 malloc()이나

free() 등의 함수를 사용하여 동적으로 메모리 공간을 할당 받는 heap 영역이 차지하며 함수의 지역변수 등을 담는 스택 영역은 사용자 공간과 커널 공간의 경계위치부터 아래방향으로 공간을 차지한다.

프로세스는 크게 텍스트, 데이터, 스택, 힙이라는 세그먼트로 구분하고 vm_area_struct 가 관리한다.