Xilinx Zynq FPGA, TI DSP, MCU 기반의 프로그래밍 및 회로 설계 전문가 과정

#32

2018.04.06

강사:Innova Lee(이 상훈)

학생: 김시윤

과제

1 장.

p24

386 보호모드 인텔계열 머신에만 있다.
페이징 인터럽트를 구현해서 최초버전을 만들었다.

FAS 프리 소프트웨어 프리소프트웨어는 다 공개 이프로그램으로는 돈을 받으면 안됨
리처드 가 만든 GNU 게임할라고 만듦

오픈소스 오픈소프트웨어

24 페이저 세번째문단 1 번째줄에서 3 번째 줄까지 외우세요.
프리소프트웨어로 표기 하지 말아라 !!!!!!!!!!
GPL 이라 써야함 회사랑 상담. (GPL 은 소스코드 공개해야함)

25 페이지 3 번째문단 두번째줄

리눅스는 유닉스 계열의 운영체제다.
데니스리치 캠톰슨 C 언어 만들음 게임을 만들면서 유닉스를 만듦

25 페이지 마지막 문단 마지막줄

생명을 제공하는 task - 장소를 제공하는 파일이다.
Task or 프로세스
프로세스가 실행되지 않으면 디스크
프로세스가 실행되서 메모리 올라가면 태스크

페이지 27

마크라는 마이크로 커널 -> 마이크로 커널 일부분을 포함 = 디바이스 드라이버
마이크로 커널은 레고처럼 탈부착이 가능함. usb 꽂았다 뺐다 하는 방식이 마이크로 커널 방식.

마이크로커널 의 반대 -> 모놀리식 커널
두번째 문단에 마지막에서 세째줄.
리눅스는 모놀리식인데 유일하게 디바이스드라이버만 마이크로식.
모놀리식은 탈부착이 안됨 통짜임 하나로 다 때려박음.
이 두가지를 하이브리드 방식으로 체텍한 리눅스는 사용하는 장치 여러개면 상황에 따라 붙이고 뗄수 있음.
그래서 네트워크 장비에서 리눅스를 쓰는것.

---

29 페이지

리눅스 장점을 요약한 구간

사용자 임의대로 재구성이 가능하다. - 커스텀라이징
(최적화)열악한 상황에서도 하드웨어자원을 적절히 사용할수 있다.-리눅스는 모놀리식이라 필요한건 마이크로 커널방식으로 탈부착 가능.구글서버 리눅스.
커널 크기가 작다.(윈도우 보다는 작다)
완벽한 멀티유저 밀터태스킹 시스템
뛰어난 안전성 (네트워크 장비에 들어간다는게 검증이 된거)
빠른 업그레이드.
강력한 네트워크 지워
풍부한 소프트웨어 GNU 재단 덕분에
사용자를 위한 여러가지 공개문서.

29 페이지 두번째 문단
상용화가 들되어있다 (옛날말)
믿을수 있나 (레이더 기지국 네트워크)

신뢰성

29 페이지 마지막 문단 첫번째줄

안된다면 내가 만들면 그만이다 -씨언어 프로그래머가 갖는 올바른 자세

2 장

cpu 를 관리한다는건 task 메모리는 page 디바이스는 드라이버 관리한다나느거
사용자가 사용하는거 시스템콜!!
사용자 입장에선 시스템콜 가지고 처리해야한다!.

fd open , sock open 두개 동작이 다른데 구동자체는 똑같다.
소켓은 원격의 피시에서 갖고오고 파일은 파일에서 갖고온다
다른동작인데 인터페이스가 동일하다 이걸 가능하게 해주는게 디바이스 드라이버.

34 페이지

운영체제는 자원공간 task_struct 자원이다.
int main 이건 유저영역.
우리가 만든 프로그램 이 자원이다.
프로그램 만든 파일 자원.
커널이 관리하면 자원

2 번째 문단
운영체제 관리하는 자원은 물리자원 추상적자원
추상적자원이 소프트웨어
추상적인녀석 관리가 어려워 눈에보이지 않아서.

추상자원을 관리한다는게 예를들어 하드웨어를 회로로 관리하면 정해진동작밖에 못해 (정해진것만 명령체계로 내려가는 구조)
소프트웨어로 관리하면 정해진것도 할수있지만 뭔가 만들어서 새로운걸 할수있다.
cpu 추상화 프로세스 (메모리에 올라가서 기계어코드를 돌려 cpu 가 기계어 코드 파악을해야하는데 프로세스들끼리 경쟁을 해야함 시피유를 차지하기 위해 그래서 cpu 자원을 가질때마다 동작하니까 시피유로봐도 무방)

추상자원
cpu 를 추상화시킨 task
세그먼트 가상메모리
페이지 물리메모리
파일 디스크
네트워크 추상화시킨 통신프로토콜 패킷.

35 페이지

물리적인 자원이랑 매칭이 안되는데 존재하는거.
stat 시스템콜로 얻어온거 파일종류 접근권한 등 이런건 물리자원이랑 1:1 로 매칭이 안됨
순수하게 소프트웨어 자원도 있다.

35 페이지 두번째 문단 읽어봐라

35 페이지 세번째 문단

task 생성 -> fork , pthread_create
실행 -> exec
상태전이 ->런큐 웨잇큐
스케쥴 -> 타임슬라이스 (시간얼마나 줄건지)
시그날 -> 시그날 처리
프로세스간 통신 -> 프로세스는 기본적으로 서로 메모리를 공유할수 없어 메모리를 공유하기 위해 쉐어드 메모리 파이프 메시지큐 를 할수 있다. 또 소켓도 네트워크긴 하지만 아이피시다 프로세스간 통신 가능이다.
그리고 thread 는 메모리를 공유하는데 크리티컬섹션이 문제가 되었다.그거때매 뮤택스랑 세마포어가 존재했다.

3 번째줄
메모리 관리자는 물리메모리 -디램을 관리
가상메모리 관리 - 스택 , 힙 , 데이터 , 텍스트 관리
페이지 부재결함 - 세그먼테이션 폴트 (os 의 권한을 침범했기 때문에 커널도 자기가 사용할

메모리가 필요한데 커널도 프로그램이기 때문 메모리메니지먼트에서 메모리를 받아옴 애초에 커널한테 메모리 할당이 안되있다. 그래서 메모리에 접근 세그먼테이션 폴트 결구 ㄱ 페이지 폴트 페이지 폴트 핸들러 페이지 할당 이후 다시 접근.

35 페이지
다섯번째!
파일시스템의 파일의 생성 opensystemcall o_creat
접근제어
아이노드 관리 - 디스크 블록의 위치를 알수 있는거. 실제 데이터가 어디있는지 관리하기위해서.
디렉토리 관리 하는게 - 디 앤트리랑 패스에서 관리함
수퍼블록 관리 - 루트파일시스템 위치를 알아오는 녀석

네트워크 - TCP/IP 통신 때문. 아이피랑 맥주소 알아야통신가능 여차하면 포트번호도
디바이스 드라이버 - 실제 장치 자체를 구동시키는 소프트웨어 실질적으론 펌웨어 OS 상에서 동작하냐 아니냐 .

운영체제가 뭐를 위해서 자원을 관리하냐 4 번쨰문단
사용자에게 서비스를 제공한다.
커널이 우리가 오픈하면 위에 정보를 가져와주는거.
우리는 요청만한다. 요청하면 복잡한 작업을 커널이 다 처리 해줌.
그 서비스를 제공해주는게 시스템콜
시스템콜을 사용하는방법을 배운거고 커널내부동작은 우리가 아직 모른다.

36 페이지
blackfin -오디오 비디오 용
openrisc - 리스크 플랫폼을 오픈 플랫폼으로 만들어? 오픈소스의 리스크 소스? 이걸 공부하면 우리만의 시피유 설계가능.
x86 인텔 시피유들.
c6x-ti dsp---
ia64 - 인텔의 멸망한 viw arch? 이거때매 그래픽카드가 엄청 떴다.
microblaze - fpga 에서 구동되는 가상시피유 소프트웨어 시피유
sparc= 슈퍼컴터용 시피유
arm - --
powerpc - 자동차에 들어감
arm64 - 앎도 64 비트가 나왔다. 구동된다.---
avr32 - avr 도 리눅스 건들었다 coretax 만듦
nios2 - altera fpga 인텔이 인수함 인텔이 밀고있는 arch

arm 은 리스트가 많은데 그이유가 하위호환을 안해서 그럼 그래서 종류가 많음. 인텔인 x86 은단순한데 arm 은 많음

arm -> mach-exynos(samsung handphone) , omap1 omap2(TI) , zynq(자일린스), s3c24xx(삼성),keystone(TIDSP)
plat-samsung(플랫폼 커널에 올라온건 대단한거),lpc18xx lpc32xx(차량용), bcm(라즈베리파이),davinci(TI, black Box 용 스마트 TV CCTV 비디오시장),stm32(stm & arm coretaxR in linux),tegra(Nvidia 사용하는 리눅스 플랫폼)

arch 디렉토리 시피유 모델이 따라서 바껴야하는 코드들이 들어있음 contaxt switching, boot loader
arch 에있는 커널은 시피유 하드웨어 관리 커널에 있는 커널은 순수하게 소프트웨어 관리. 페이징하는 코드가 알키에 mm 페이징 하는데 소프트웨어적인 부분은 커널에 mm
arch/include lib 코드할수있는 인클루드랑 라이브러리.
tools 커널에서 사용할수 있는 툴들.
block -> cpu 여러개 있는 운영체제 만들때 어떤한걸 해야하나? 블록파일 캐릭터파일 소켓 파이프 파일, 블록 디바이스 드라이버 관련된거 들어있음 블록이 규모가 커서 들어있음.
init - 커널의 시작코드가 들어있다. start_kernel 에 하드웨어 초기화
vi -t start_kernel 의 9 번
init 에 커널에 초기화 함수

samples 샘플이있다
usr 유저가 사용할때 관계된 정보
certs 암호랑 관련된게 들어있음
Documentation 커널 개발에 대한게 들어있음.
ipc - ipc 관련된 매커니즘이 다 구현되어있음. 세마포어 메시지큐 뮤택스 등
scipts 커널에 필요한 스크립트?
virt -- 운영체제에 의해 운영체제를 올릴수 있게 해줌 운영체제를 프로세스처럼 동작시키게 해준다.

drivers 드라이버에는 디바이스 드라이버에 관련된 것들이 엄청 많다.
iio - 오실로스코프 만들수 있다.

mm = 메모리 메니지먼트 가상메모리 관리 물리메모리 관리하는 페이징기법이 구현되어있음

security = 보안과 관련된거

firmware - 빌트된 펌웨어랑 통신하기위해
net - network 와 관련된거
soud - 오디오 관련된거.

오디오 - ALSA
비디오 - V4L2
crypto 암호화 복호화 다들어있어 암호학
cert 는 암호학을 사용하는거

fs 파일 시스템 -> 파일과 관련된 작업들을 싹다함.
kernel 커널과 관련된 소프트웨어 대표적으로 스케줄???
38-39 설명

39 페이지

리눅스 커널 컴파일.
DSP FPGA 할때 할꺼다.
DSP 커널 컴파일 오래걸림
나중에 할거다.

44 페이지
3 번째 문단.

M 은 모듈 M 이라고 나와있으면 탈부착 가능
드라이버 탈부착 가능.

45 페이지
두번째줄
m

n 이 들어가면 모놀리식 y 나

3 장

프로그램을 실행하면 프로세스가 살아남
살아나면서 제어권을 가져가고 구동하면서 작업

태스크에 라이프 사이클
생명을 준다는 얘기가 이거고
변화를 준다는건
이그젝을 하면서 쇼크를 줘서 변화를 시킨다.
생명준다 - fork or pthread

변화준다 = exec

문맥교환 - contaxt switch

프로세스와 thread 전부 태스크.
thread 는 프로세스가 될수도 있고 안될수도 있다 thread 가 더 넓은 범주

50 페이지 첫번쨔 문단

피아이디와 티지아이디가 같으면 프로세스
티지아이디만 같고 피아이디가 다르면 같은 스레드 그룹 구성원

포크를 하면
서로 다른 피아이디 티지아이디가 생기는데
부모 피아이디 티지아이디 같고

자식 피아이디 티지아이디가 같고
부모자식서로 아이디 는 다름

50 페이지 두번째문단.

file a.out
readelf -h a.out

50 페이지 마지막 문단.

실행파일은 메모리에 올라가야 프로세스가 된다.

51 페이지

프로세스가 어떻게 생겼을까?
사용자 입장에서 생김새를 논할때는 가상메모리를 얘기한다.

가상메모리는 물리메모리가 아니고 짝퉁이다.

첫번쨔 문단에 세번째줄

프로세스 별로 주어지는 가상주소공간 이것도 ??

32 비트는 1:3 을로 나누고있고 (총 4 기가)

64 비트는 사용자공간이 값이 이상해? 1:1 로 보는게 좋다?

그림 3.2

52 천번쨔문단 두번째줄

리전 -> 세그먼트(가상메모리) 즉 가상메모리 주소를 얘기한다.

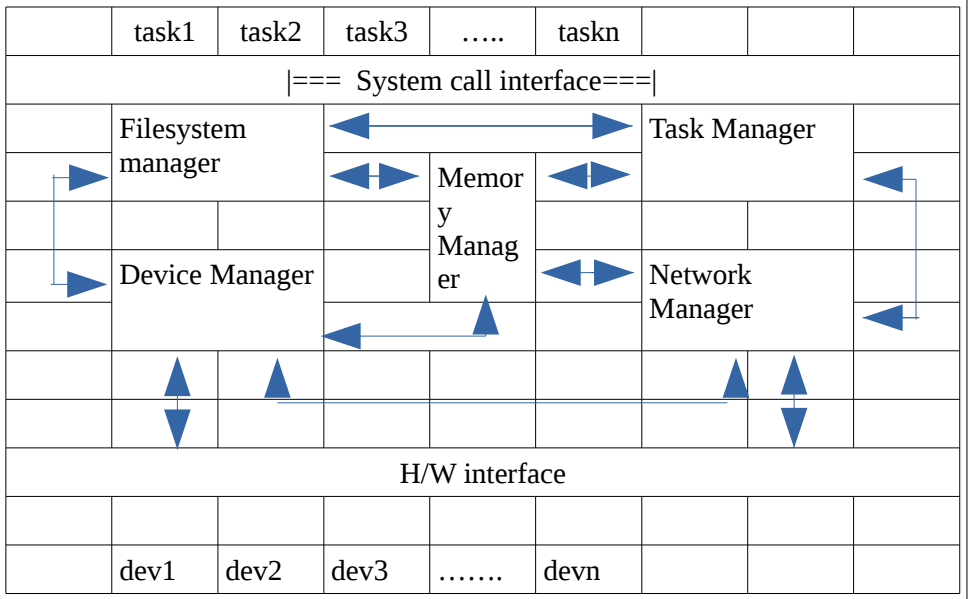
지역변수 스택

malloc or free 는 동적 heap

텍스트 데이터 힙 스택 세그먼트(리전)
관리는 vm_struct 가함

53 페이지 보면
이때 각 영역 리전을 세그먼트로 바꿔라.

그림 2.1



System Call interface – user 역역
유저가 명령하는 것들은 다 시스템콜이다.
유저는 task_struct 로 관리된다.

5 개 file system , task manager, Device manager memory manager network manager 로 묶여있는데 서로가 서로를 가르킨다. 즉 하나라도 없으면 제도로 동작이 불가능하다.

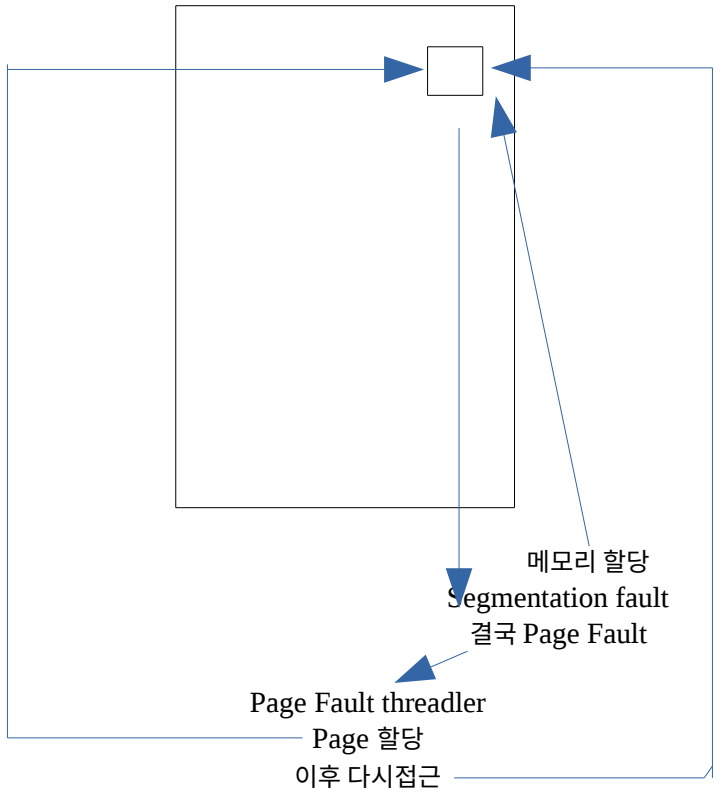
태스크매니저에서 스케줄로 메모리에 프로그램이 올라간다.

파일시스템은 파일이 디스크상에 어디있는지를 관리한다.

네트워크는 원격 관리

dev = 물리적인 장치.

커널 메모리



pid = 프로세스 아이디
tgid = thread group ID
ex) 삼성 회장 이건희 는 Pid Tgid 동일
삼성 사원들은 pid tgid 가 다름.

Tgid = pid => 프로세스 (스레드 리더)
다르면 스레드

fork() => pid = tgid