Xilinx Zynq FPGA, TI DSP MCU 기반의

프로그래밍 및 회로 설계 전문가

강사 이상훈

(Innova Lee)

Gcccompil3r@gmail.com

학생 김민호

minking12@naver.com

5번
사용자 임의대로 재구성 가능
(커널에 들어가서 원하는 대로 소스코드 수정가능)

열악한 환경에서...
(리눅스 커널이 가볍고 , 많은 사람들이 지속적으로 개발하여 열약한 환경에서도 잘동작함.)

커널크기가 작다.
(최적화)

멀티유저,멀티태스킹 시스템
(RT 스케줄링(실시간 처리) 방식 채택하여 멀티 태스킹이 효율적이다.)

안정성
(하도 많은 사람들이 지속적으로 유지보수하여 안정성이 뛰어남.)

네트워크
(tcp/ip 스택이 잘되어있다.)

풍부한 sw
(GNU 스톨만 정신에 입각하여 많은tool들이 개발되어 있다.)

7번
page fault 발생시 운영체제가 어떻게 동작하는지 ?

(page fault 는 가상메모리를 물리 메모리로 변환중 물리 메모리에 할당된 페이지가 없을 경우 발생.
페이지에 대한 쓰기 권한이 있을 경우 -> page fault handler 구동 하여 페이지 할당 -> 저장해놨던 ip를 복원하여 다시 루틴을 구동,
쓰기 권한이 없다면 Segmentation Fault를 발생시킨다.)

14번
스케줄링 방식중 Deadline
(지금 당장 급한것부터 처리하라.)

17번 18번
(부모가 fork를 하면 자식은 최우선으로 부모쪽으로 가서 런큐쪽으로 빠지고 그 다음순위는 기존 있던곳 그 다음 순위는 가까운 곳이다. -> Cache Affinity)

19번
(뱅크 1개 -> UMA
뱅크 2개이상 -> NUMA
뱅크 관리하는 구조체가 NODE
노드는 pglist_data 구조체)

20번
(실시간 처리 1~99 ,
동적처리 100~139)

21번
ZONE HIGHMEM
(커널영역이 1:1 이상 대응 초과했을때 하이멤이 해준다.)

23번 24번
(레알 마드리드 정상용 학우가 너무 잘가르쳐줘서 확실하게 이해했습니다. 근데 책에 그림으로 설명해야지 잘 할수있어서 패스 하겠습니다 .ㅎ.....)

25번 26번
가상메모리 구조인 stack ... 등등 어디에 기록되는가 ?
(task_struct->mm_struct->vm_area_struct에서 속성이 같은것들끼리 모아서 관리 한다.)

27번
(ELF Header)
페이징 이론을 보면 알수있다!

28번
커널이나 유저 영역이나
운영해주는 sw가 있어야 작동하니까 스택이 필요하다(c언어관점)

29번
vm 에서 pm 으로 변환시키는 페이징
(페이지 테이블을 10/10/12 vm 의 주소로 쪼개서 인덱싱을 한다.)

30번
MMU
(HAT 가상메모리 공간을 실제 메모리 공간으로 변환 ,
TLB 가상메모리 주소와 대응되는 물리 메모리 주소를 cashing 한다.)

31번
디스크 블록 -> 4kb 이 아니라
섹터 -> 512 byte

32번
Character 디바이스 드라이버를 작성할 때 반드시 Wrapping 해야 하는 부분이 어디인가 ?
(Task 구조체에서 부터 연결된 부분까지를 쭉 이어서 작성하라)

task_struct->files_struct->file->file_operations

33번
open 시스템콜 했을때 커널에서 어떤 함수가 작동
(sys_open())

34번
task_struct 에서 super_block의 역할
('/' fs에서 원하는 파일을 찾을때)

35번 VFS
(ext 방식이나 inode 방식이나
수많은 함수를 일일히 다 파고 들어서 실행시킬수 없으니 이놈이 대신 해준다.)

36번 37번
(내부 인터럽트, 외부 인터럽트
내부는 fault ,trap, abort)

40번 41번
(인텔 : sys_call_table -> ax
ARM: __vectors_start + 0x1000
->r7)

42번
페이징 이론에서 페이지 디렉토리는 mm_struct 의 어떤 변수가 가지고 있는가 ?
(pgd)

43번
(인텔-> CR3
ARM -> CP15)

44번
커널내부에서도 메모리 할당이 필요?
(커널도 프로그램이니까 필요 ..)

45번
메모리 불연속적 할당기법
(vmalloc)

46번
메모리 연속적 할당기법
(kmalloc)

48번
module_init 함수는 insmod명령어로 디바이스 드라이버 모듈 부착시 동작

49번
module_exit 함수는 rmmod 명령어로 드라이버 모듈을 탈착시킬때 동작한다.

50번
thread_union
(thread_info
*task, cpu_context_save)

51번
(task_struct -> files_struct 내에 path 내에 디엔트리내에 inode 구조체에 존재하는 i_rdev 변수에 저장한다.)

55번
OoO
(데이터 의존성이 존재하게 되면 stall 이 발생하는데 이걸 최소화하기 위해 앞서 실행했던 코드와 의존성이 없는 코드를 찾아서 의존성이 있는 코드위로 끌어올려서 실행하는 방식.)

57번
CISC , RISK 대한 차이점
(cisc 는 여러 기능이 있어서 전력소비가 크다)

58번
인텔계열은 컴파일러 없이도 스케줄링을 HW 만으로 할수있는데 무엇인가 ?
(Dynamic Instruction Scheduling)

59번
파이프 라인 깨지는 경우
(분기 발생 -> jmp or branch)

60번
cpu는 저마다 이것을 가지고 있다. 컴파일러 HW 개발자입장에서 이것을 고려하여 설계
(ISA : instruction set architecture 명령어 집합)

61번
하이퍼 쓰레딩
(cpu 성능이 4배까지 좋아짐 ?)

68번
현재 삽입된 디바이스 드라이버 리스트를 보는 명령어
(lsmod)

69번
시스템콜 메카니즘
(유저가 커널영역으로 요청하는것인데 실질적으로 커널이 알아서 수행한다.)

70번
프로세스와 vm 의 관계
(프로세스는 자신의 고유한 공간으로 4gb를 가지고 있고, 실제 이 공간을 모두 사용하지 않으며 Demand Paging 에 따라 필요한 경우에만 실제 물리 메모리 공간을 Paging 메카니즘을 통해 할당 받아 사용한다.)

74번
COW
(copy on write 실직적인 쓰기가 발생하는 시점에 복사를 하겠다는 뜻.)

80번
리눅스에서 fd는?
(리눅스 커널에 존재하는 task를 나타내는 구조체인 task_struct 내에 file_struct 내에 file 구조체에 대한 포인터가 fd_array 이다. 거기서 우리가 시스템프로그래밍에서 얻는 fd는 바로 이 fd_array에 대한 index 다.)

82번
프로세스는 최소메모리 관리하기 위한 mm, fd_array, signal 을 포함하는데 그 이유는 ?
(자신이 실제 메모리 어디에 위치하는지 정보가 필요, 디스크의 어떤 파일이 메모리에 로드되어 프로세스가 되는지 정보가 필요 ... 프로세스들 간에 signal 주고 받을 필요가 있기 때문에 signal 필요.)

87번
클라우드 기술의 핵심인 os
os 가상화에서 핵심 3가지
(cpu 가상화 , 메모리 가상화, IO 가상화)

93번
Critical Section
(프로세스나 스레드가 동시 다발적으로 접근하여 정보를 공유할 수 있는 영역을 의미. 연산이 아직 끝나지 않은 상태로 연산을 수행하게 되면 해당 연산이 의미가 없어질 수 있다는 문제점이 존재.)

99번
ARM 에서 시스템콜 사용할때 사용하는 레지스터는 ?
(r7)