



EDDI

Electronic Design
Development Institute

에디로봇아카데미

임베디드 마스터 Lv2 과정

제 1기

2022. 07. 16

손표훈

CONTENTS

- 멀티 프로세스
 - 프로세스란?
 - 부모 프로세스와 자식프로세스
 - Copy On Write(COW)
 - 프로세스와 쓰레드의 차이
- SIGNAL
 - SIGNAL이란?
 - wait 함수

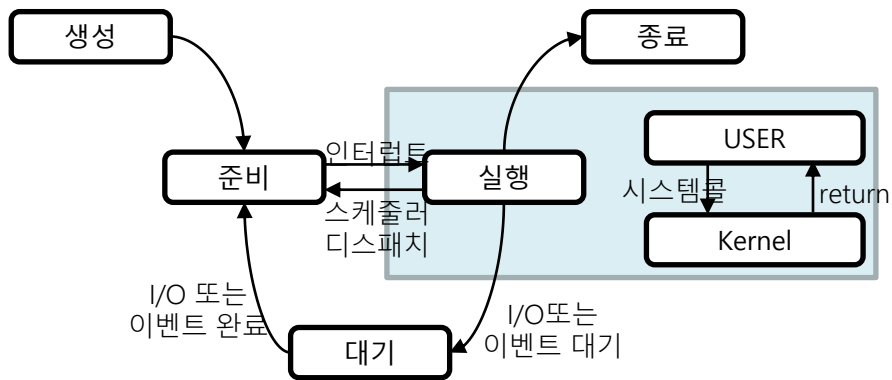
멀티 프로세스

➤ 프로세스란?

- 프로세스는 실행 중인 프로그램이며, 생성, 준비, 대기, 실행, 완료 등의 다양한 상태를 가지고 있다
- 리눅스에서는 **시분할 스케줄링 알고리즘(Completely Fair Scheduler-CFS)**에 의해 프로세스를 제어 한다
 - * CFS는 RB트리 구조로 구성 되어 있다
- 프로세스간에 서로 공유하는 자원이 없다. 즉, 먼저 CPU를 점유하는 프로세스가 CPU의 자원을 사용한다
 - *프로세스는 CPU의 추상화다!

포인터	상태
프로세스 ID	
프로그램 카운터	
버퍼 포인터	
레지스터	
메모리 제한	
열린 파일 목록	
자식 링크	
....	

Process Control Block

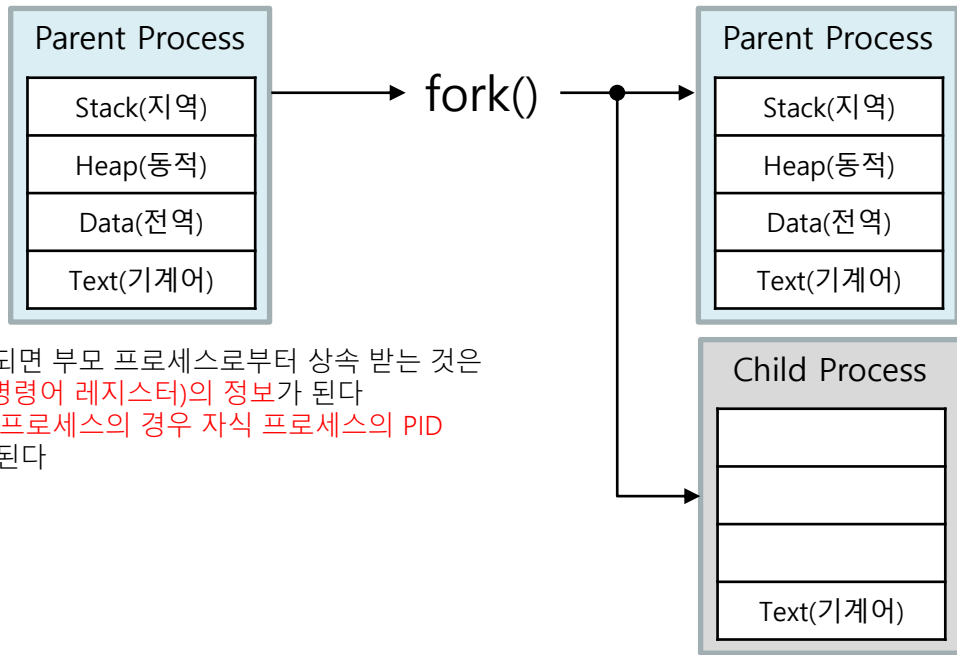


*사물인터넷을 위한 리눅스 프로그래밍 with 라즈베리파이(서영진 지음) 참고

멀티 프로세스

➤ 부모 프로세스와 자식프로세스

- 자식 프로세스는 부모 프로세스의 분신과 같다
- 자식 프로세스가 생성 될 때 부모 프로세스의 **메모리 구조를 상속 받지만 서로 분리되어 있다(즉 각각의 독립된 메모리 구조를 갖는다)**
- 즉 또 다른 프로세스가 하나 더 생성되는 것
- 리눅스에서 fork 함수를 통해 자식 프로세스를 생성 할 수 있다
- fork 함수 이전을 통해 자식 프로세스의 메모리 구조는 다음과 같이 복사 된다



- 우선 자식프로세스가 생성되면 부모 프로세스로부터 상속 받는 것은 **Text영역과 현재 RIP(현재 명령어 레지스터)의 정보가 된다**
- fork함수의 리턴 값은 **부모 프로세스의 경우 자식 프로세스의 PID**
자식 프로세스의 경우 0이 된다

멀티 프로세스





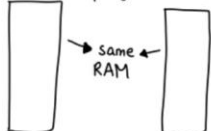

➤ Copy On Write(COW)

<https://wizardzines.com/comics/copy-on-write/>

JULIA EVANS
@b0rk

copy on write

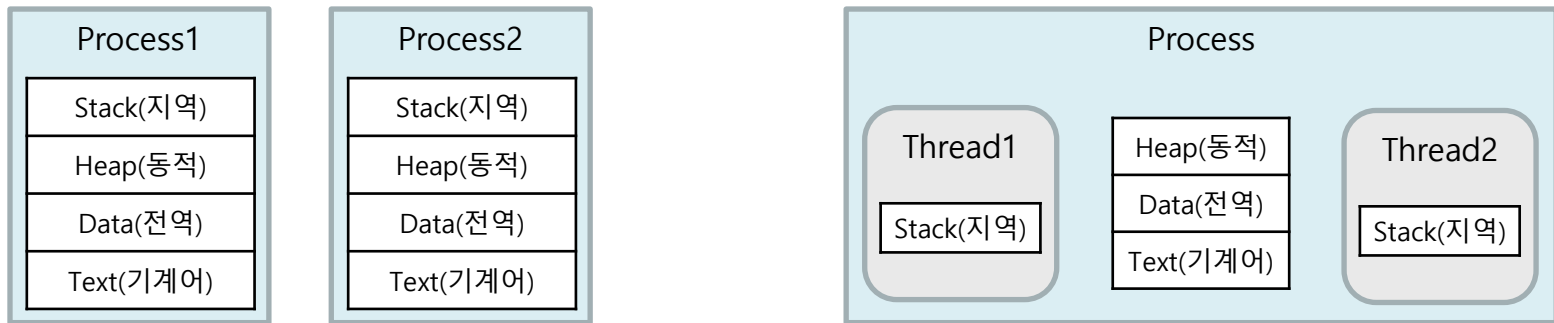
19

<p>On Linux, you start new processes using the <code>fork()</code> or <code>clone()</code> system call.</p> <p>calling <code>fork</code> creates a child process that's a copy of the caller</p> <div> parent</div> <div> child</div>	<p>the cloned process has EXACTLY the same memory.</p> <ul style="list-style-type: none">→ same heap→ same stack→ same memory maps <p>if the parent has 3GB of memory, the child will too</p>	<p>copying all that memory every time we fork would be slow and a waste of RAM</p> <p>often processes call <code>exec</code> right after <code>fork</code>, which means they don't use the parent process's memory basically at all!</p>
<p>so Linux lets them share physical RAM and only copies the memory when one of them tries to write</p> <div> I'd like to change that memory</div> <div> okay! I'll make you your own copy!</div>	<p>Linux does this by giving both the processes identical page tables.</p> <div></div> <p>but it marks every page as read only.</p>	<p>When a process tries to write to a shared memory address:</p> <ol style="list-style-type: none">① there's a page fault② Linux makes a copy of the page & updates the page table③ the process continues, blissfully ignorant <div> ..It's just like I have my own copy</div>

멀티 프로세스

➤ 프로세스와 쓰레드의 차이

- 프로세스와 쓰레드의 차이는 메모리 공유 여부이다
- 프로세스는 하나의 독립적인 프로그램이라고 하며, Task라고도 한다
- 프로세스는 CPU를 점유하고 멀티 프로세스 환경에서 스케줄링에 의해 프로세스의 CPU 할당 순서가 정해진다
- 즉, 프로세스는 CPU의 모든 자원을 다 사용한다(프로세스는 CPU의 추상화)
- 프로세스는 서로 메모리를 공유하지 않기 때문에 IPC(Inter Process Communication) 방식을 이용하여 정보 공유를 할 수 있다



- 쓰레드는 Stack영역만 개별적으로 할당 받고 DATA, HEAP, TEXT영역은 공유 한다
- 같은 프로세스 안에서 여러 쓰레드들은 HEAP과 DATA 영역을 공유할 수 있다

SIGNAL

➤ SIGNAL이란?

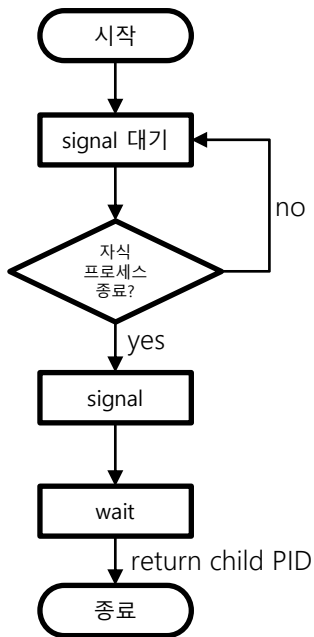
- SIGNAL은 system call로 유일한 소프트웨어 인터럽트이다
- 특정 이벤트(프로세스 종료, 비정상 종료, 비정상적인 기계어 명령 등)가 발생했을 때 인터럽트가 발생한다
- 리눅스에서 주요 시그널은 다음과 같다

번호	시그널 이름	발생 및 용도			
1	SIGHUP(HUP)	-hangup 시그널; 전화선 끊어짐 -로그아웃과 같은 터미널에서 접속이 끊겼을 때 보내지는 시그널 -데몬 관련 환경 설정 파일을 변경시키고, 변화된 내용을 적용하기 위해 재시작할 때 이 시그널이 사용됨	14	SIGALRM(ALRM)	-경보(alarm) 시그널; alarm(n)에 의해 n초 후 생성됨 -알람 타이머 만료 시에 사용
2	SIGINT(INT)	-interrupt 시그널; Ctrl+C; 실행을 중지 -키보드로부터 오는 인터럽트 시그널로 실행을 중지시킨다.	15	SIGTERM(TERM)	-일반적으로 kill 시그널이 전송되기 전에 전송된다. -잡히는 시그널이기 때문에 종료되는 것을 트랙할 수 있다.
3	SIGQUIT(QUIT)	-quit 시그널; Ctrl+\ -사용자가 터미널에서 종료키를 누를 때	16	SIGKFLT	코프로세서 스택 실패
4	SIGILL(ILL)	잘못된 명령	17	SIGCHLD(CHLD)	프로세스 종료시 그 부모 프로세스에게 보내지는 시그널
5	SIGTRAP(TRAP)	트랩 추적	18	SIGCONT(CONT)	STOP 시그널 이후 계속 진행할 때 사용; 정지되지 않은 경우 무시됨
6	SIGTIO(TIO)	-I/O 명령 -Abort(비정상 종료) 함수에 의해 발생	19	SIGSTOP(STOP)	정지 시그널; SIGSTP과 같으나 잡거나 무시할 수 없음
7	SIGBUS(BUS)	버스 에러	20	SIGTSTP(TSTP)	키보드에 의해 발생하는 시그널로 Ctrl+Z로 생성된다; 터미널 정지 문자
8	SIGFPE(FPE)	부동 소수점 에러	21	SIGTTIN	백그라운드에서의 제어터미널 읽기
9	SIGKILL(KILL)	-무조건적으로 즉시 중지한다. -Kill, 실행 중인 프로세스를 강제 종료할 때 사용	22	SIGTTOU	백그라운드에서의 제어터미널 쓰기
10	SIGUSR1(USR1)	사용자 정의 시그널1	23	SIGURG	소켓에서의 긴급한 상태
11	SIGSEGV(SEGV)	-세그멘테이션 위반 -Segmentation Violation, 메모리 액세스가 잘못되었을 때 발생	24	SIGXCPU	CPU 시간 제한 초과 setrlimit(2) 매뉴얼 페이지 참조
12	SIGUSR2(USR2)	사용자 정의 시그널2	25	SIGXFZ	파일 크기제한 초과 setrlimit(2) 매뉴얼 페이지 참조
13	SIGPIPE(PIPE)	-읽으려는 프로세스가 없는 데 파이프에 쓰려고 함 -종료된 소켓에 쓰기를 시도할 때	26	SIGVTALRM	가상 시간 경과 setitimer(2) 매뉴얼 페이지 참조
			27	SIGPROF	프로파일링 타이머 경고. setitimer(2) 매뉴얼 페이지 참조
			28	SIGWINCH	윈도우 사이즈 변경
			29	SIGIO	기술자에서 입출력이 가능함. fcntl(2) 매뉴얼 참조

SIGNAL

➤ wait 함수

- 부모 프로세스가 자식 프로세스의 종료 상태를 얻기 위해서 wait 함수를 사용한다
- 동작은 아래와 같다



- wait 함수는 blocking 동작이므로 해당 함수가 끝날 때 까지 다른 연산들을 진행하지 못한다
- 여러 개의 자식들이 생성/종료 되는 멀티 프로세스 환경에서 비정상 종료된 자식들이 동시 다발적으로 발생할 때 하나의 wait 처리로 인해 다른 비정상 종료된 상황을 인지하지 못하는 예외 상황이 발생할 수 있다
- 이를 동기처리 방식이라 한다
- 위 와 같은 상황을 방지하기 위해 Non-blocking 처리를 하게 되고 이를 bottom half 처리라 한다

```
10 void term_status(int status)
11 {
12     if (WIFEXITED(status))
13     {
14         printf("(exit)status: 0x%x\n", WEXITSTATUS(status));
15     }
16     else if (WTERMSIG(status))
17     {
18         printf("(signal)status: 0x%x, %s\n",
19             status & 0x7f, WCOREDUMP(status) ? "core dumped" : "");
20     }
21 }
22
23 void my_sig(int signo)
24 {
25     int status; blocking처리
26
27     while(wait(&status) > 0)
28     {
29         term_status(status);
30     }
31 }
```

```
10 void term_status(int status)
11 {
12     if (WIFEXITED(status))
13     {
14         printf("(exit)status: 0x%x\n", WEXITSTATUS(status));
15     }
16     else if (WTERMSIG(status))
17     {
18         printf("(signal)status: 0x%x, %s\n",
19             status & 0x7f, WCOREDUMP(status) ? "core dumped" : "");
20     }
21 }
22
23 void my_sig(int signo)
24 {
25     int status; non-blocking처리
26
27     while(waitpid(-1, &status, WNOHANG) > 0)
28     {
29         term_status(status);
30     }
31 }
```

* WNOHANG은 기다리는 PID가 종료되지 않아서
즉시 종료 상태를 회수 할 수 없는 상황에서 호출자는 차단되지 않고 반환 값으로 0을 받는다