프로세스와 잠금

1871005 강예준

한성대학교 HANSUNG UNIVERSITY

유튜브 주소 : https://youtu.be/xixBc7sOGhQ

CryptoCraft LAB
https://crypto.modoo.at

Contents

CryptoCraft LAB

01. 프로세스의 생성

01. 프로세스의 생성

- fork() 시스템 호출
  - 부모 프로세스를 똑같이 복제하여 새로운 자식 프로세스를 생성
  - fork() 한 번 호출되면 두 번 리턴
  - 이때 자식프로세스에게는 0을, 부모프로세스에게는 자식 프로세스 ID를 리턴
  - 부모 프로세스와 자식 프로세스는 병행적으로 각각 실행을 이어감

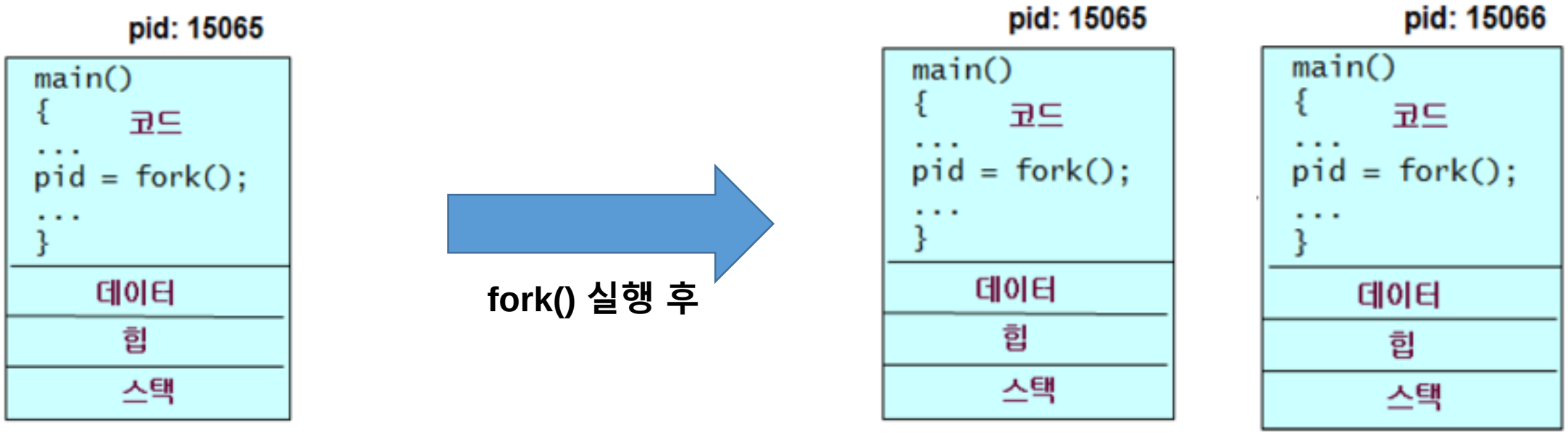
```
#include <sys/types.h>
#include <unistd.h>
pid_t fork(void);
```

01. 프로세스의 생성

- 부모 프로세스
  - fork() 함수를 호출하는 쪽
- 자식 프로세스
  - fork() 함수를 통해 새로 생성된 쪽
- 부모 프로세스와 자식 프로세스 구분하는 법
  - fork() 호출 후 리턴 값이 다르므로 리턴 값으로 구분
  - 서로 다른 코드를 실행하도록 코드를 짤 수 있음

```
pid = fork();
if ( pid == 0 )
{
    자식 프로세스의 실행 코드
 }
else
{
    부모 프로세스의 실행 코드
}
```

01. 프로세스의 생성


pid: 15065
main()
{
코드
...
pid = fork();
...
}
데이터
힙
스택
fork() 실행 후
pid: 15065
main()
{
코드
...
pid = fork();
...
}
데이터
힙
스택
pid: 15066
main()
{
코드
...
pid = fork();
...
}
데이터
힙
스택

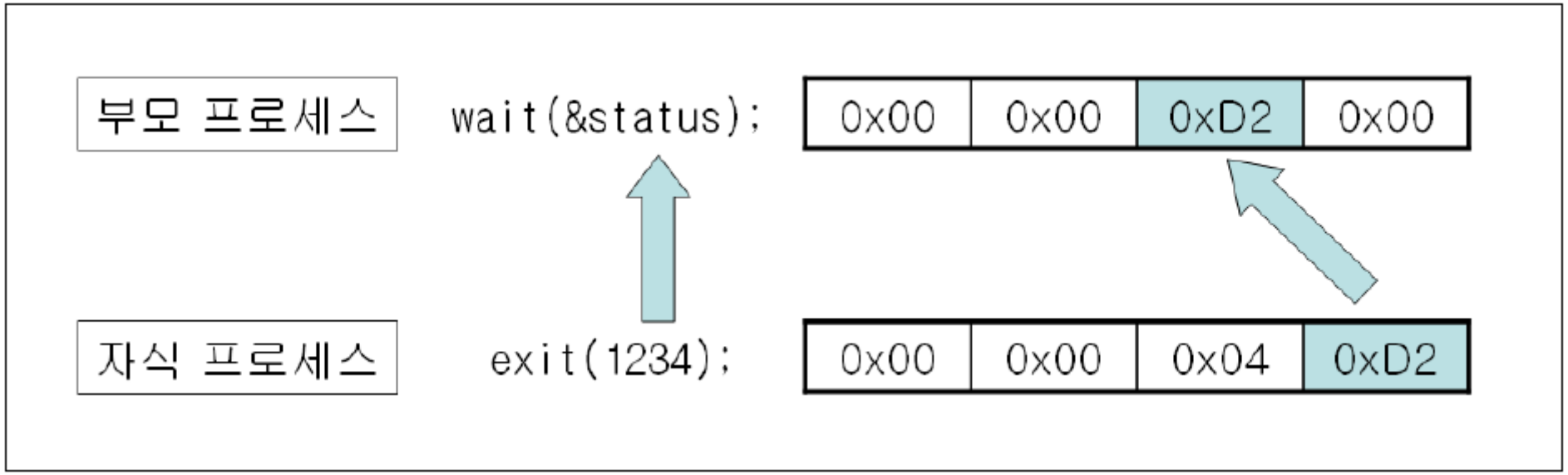
02. 프로세스 기다리기

02. 프로세스 기다리기

- wait()
  - 자식 프로세스 중의 하나가 끝날 때까지 기다린다.
  - wait 호출 이전에 자식이 종료했다면 대기 상태가 되지 않고 즉시 리턴
  - 끝난 자식 프로세스의 종료 코드가 status에 저장
  - 끝난 자식 프로세스의 번호를 리턴

- waitpid()
  - 부모 프로세스가 PID로 자식 프로세스를 지정하여 기다린다.
  - 옵션에 따라 자식 프로세스가 종료할 때까지 대기 상태가 될 수도 있고 아닐 수도 있음.

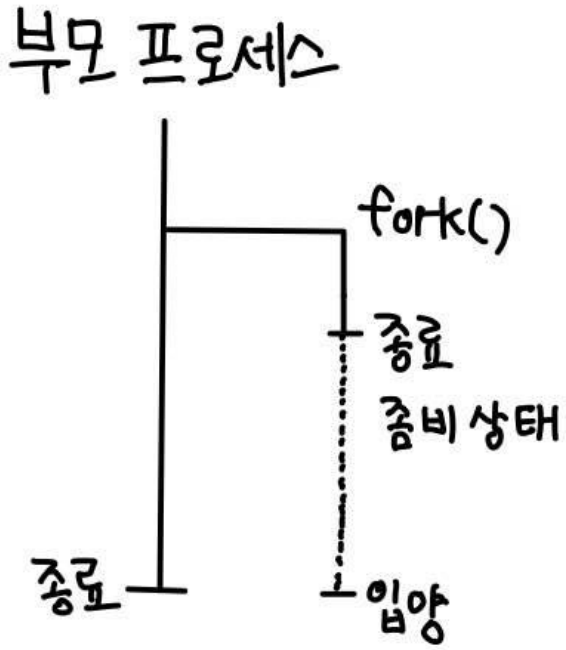
2. 프로세스 기다리기


부모 프로세스
wait(&status);
0x00
0x00
0xD2
0x00
자식 프로세스
exit(1234);
0x00
0x00
0x04
0xD2

03. 좀비프로세스와 고아프로세스

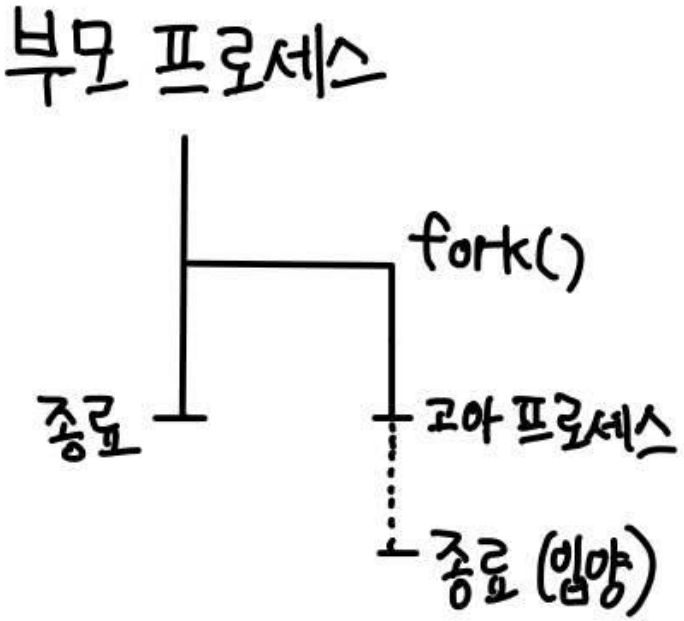
03. 좀비 프로세스와 고아 프로세스

- 좀비 프로세스 (ZOMBIE PROCESS)
  - 부모 프로세스가 wait를 수행하지 않고 있는 상태에서 자식이 종료한 경우
  - 좀비 프로세스는 CPU, Memory 등의 자원을 사용하지 않음
  - 단, 커널의 작업 리스트에는 존재



03. 좀비 프로세스와 고아 프로세스

- 고아 프로세스 (ORPHAN PROCESS)
  - 부모 프로세스가 수행 중인 자식 프로세스를 기다리지 않고 먼저 종료한 경우



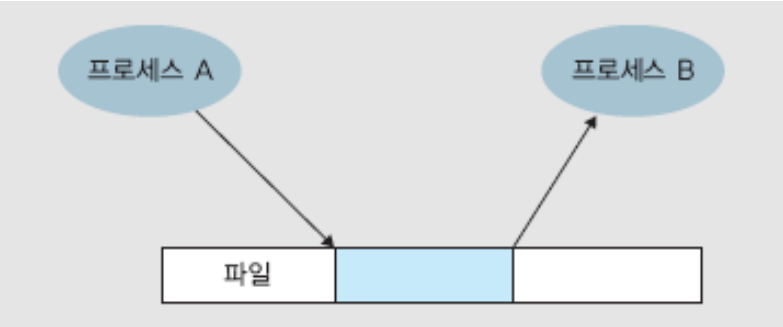
03. 좀비 프로세스와 고아 프로세스

- Init 프로세스
  - 유닉스/리눅스 시스템이 부팅 되는 과정에서 가장 먼저 실행되는 프로세스
  - PID가 1번인 프로세스
  - 좀비와 고아 프로세스는 결국 init 프로세스로 넘겨진다
  - Init 프로세스가 wiat함수를 호출

04. 파일 및 레코드 잠금

04. 파일 및 레코드 잠금

- 프로세스끼리 데이터를 주고 받는 원리
  - 한 프로세스가 파일에 쓴 내용을 다른 프로세스가 읽음



- 문제점
  - 한 프로세스가 파일 내용을 수정하는 동안에 다른 프로세스가 그 파일을 읽는 경우
  - 두 개의 프로세스가 하나의 파일에 동시에 접근하여 데이터를 쓰는 경우

04. 파일 및 레코드 잠금

문제 발생

(1) 프로세스 A가 잔액을 읽는다: 잔액 100만원

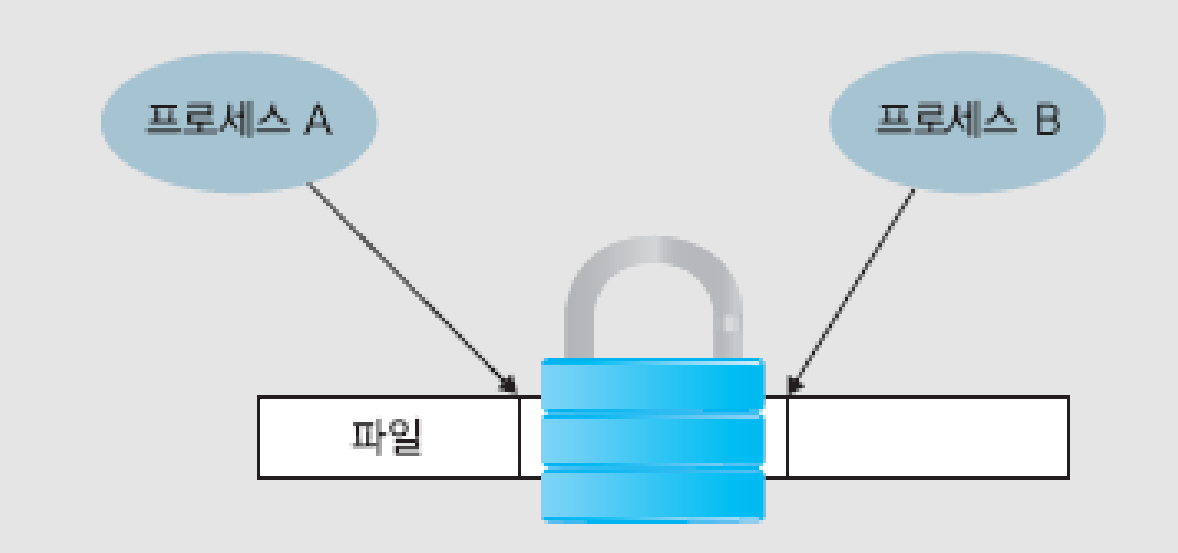
(2) 프로세스 B가 잔액을 읽는다: 잔액 100만원

(3) 프로세스 B가 잔액에 입금액 20만 원을 더하여 레코드를 수정한다: 잔액 120만원

(4) 프로세스 A가 잔액에 입금액 10만 원 을 더하여 레코드를 수정한다: 잔액 110만원

04. 파일 및 레코드 잠금

- 잠금(lock)
  - 한 프로세스가 그 영역을 읽거나 수정할 때 다른 프로세스의 접근을 제한



04. 파일 및 레코드 잠금

잠금 사용

- (1) 프로세스 A가 레코드에 잠금을 하고 잔액을 읽는다: 잔액 100만원
- (2) 프로세스 A가 잔액에 입금액을 더하여 레코드를 수정하고 잠금을 푼다: 잔액 110만원
- (3) 프로세스 B가 레코드에 잠금을 하고 잔액을 읽는다: 잔액 110만원
- (4) 프로세스 B가 잔액에 입금액을 더하여 레코드를 수정하고 잠금을 푼다: 잔액 130만원

- 예제

감사합니다!