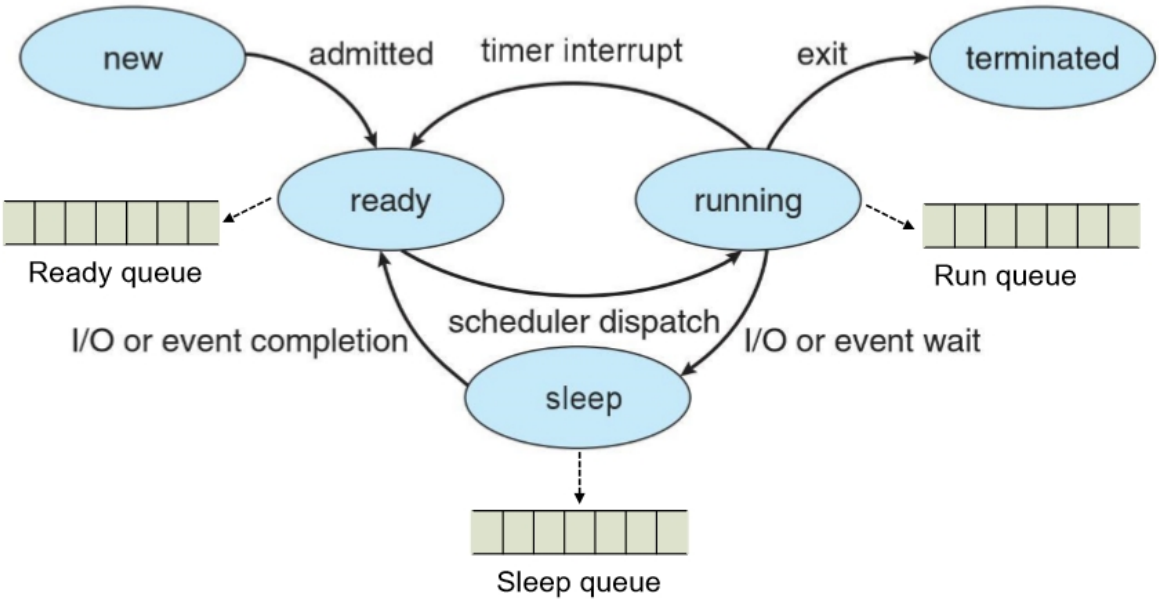
7강

Transition of process state

프로세스들이 어떤 상태를 가진다는 의미이다.



Process creation

요즘은 프로세스가 다른 프로세스를 만들 수 있다.

parant가 child를 만들 수 있게 해주는 것이다.

- 프로세스 생성을 용이하게 하기 위하여
  - 프로세스에서 다른 프로세스를 만들 수 있도록 함
    - fork(2) - system call
  - init - 최초의 가장 먼저 생기는 프로세스

    →init이 fork를 해서 다른 프로세스들을 만드는 것이다.

- Process creation by fork(2)
  - Parent process vs. child process
    - Initially child process is a duplicate of parent process.

      → 처음에는 child가 parent를 그대로 복사한다. TDS가 똑같다.
  - Some resources are shared because initially PCB is copied

- Execution
  - Parent and children execute concurrently.

    두개가 concurrently하게 실행된다.

  - Parent can wait until children terminate (SIGCHLD).

    차일드가 페런트가 멈출때까지 기다리는 기능이 있다. 그 신호는SIGHLD이다.

멀티테스킹은 프로세스가 프로세스를 만들어 실행시키는 기능을 지원한다는 의미이다.

Q. parent와 child가 concurrent하게 실행된다는게 잘 이해가 안되는데, parent와 child가 같이 스케쥴링 되면서 parent 처리가 끝난 후 child가 처리되는것처럼 내부적으로는 batch로 실행되는건가요

A. P와 C가 독립적으로 작동한다는 의미이다.

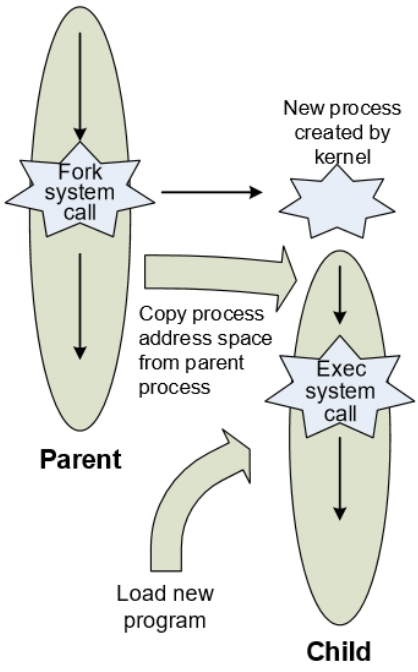
child가 P에게 끝난다고 말할 수 있는 관계도 있다.

Process creation in memory view

job과 task가 뭐냐?

task는 여러가지를 지칭할 수 있다. container, process, thread 등

task는 좀더 포괄적으로 사용되는 의미이다.



1. A new process created by fork system call

   →fork라는 시스템콜으로 복사가 된다.

2. A new process (child process) consists of a copy of the memory of the original process (parent process)
3. Exec(2): to replace the memory with a new program

   → Exec라는 시스템 콜을 사용해서 새로운 프로그램으로 만든다.

   → 메모리를 새로운 프로그램으로 교체한다.

ex)

shell이라는 프로그램이 ls를 보면 ls를 받은 후에 자신을 복사하고, 복사된 자신을 ls라는 프로그램으로 교체해 버리는 것이다.

→ 어떠한 커맨드를 넣더라도 이러한 절차를 통해 수행이 되는 것이다.

왜 이렇게 하나? 운영체제는 잘 정의된 structure를 필요로하기 때문이다.

Q. 그냥 새로운 프로그램을 로드하면 되지 왜 copy라는 정차가 필요한가?

A. RTS를 생각해봐라 이것이 프로그램을 올리는데, 이것을 하려면 커널데이터 스트럭쳐를 할당해 줘야 할 필요성이 있다. 프로세스를 만드는 하는 과정 자체가 crapt하다고 한다. 그것은 한땀한땀 만드는 것이다. 이것을 줄이기 위해서 그냥 복사를 해버리는 것이다.

물론 복사는 해야하지만, 프로세스를 만드는데 쓰는 비용을 줄이기 위함이다. 죽, 효율의 문제이다.

운영체제는 형식을 잘 지키는 것이 중요한데, 그것보다 더 중요한 것은 "성능"이다.

"성능", "성능", "성능"이다. 이것이 미덕이다.

Q. 질문이 있습니다. init 프로세스가 처음에 프로세스들을 만들고 그 프로세스들이 fork를 해서 만들어지면 init 프로세스만 craft 과정으로 생성되는 것인가요?

A. 정확하다. 부팅할때 init 프로세스를 만드는 과정이 포함되어 있다.

요즘 티비부팅하는 것과 컴퓨터 부팅하는 것은 시간차이가 많이 난다.

→ 요즘 티비는 linux가 다 들어간다. 전부 리눅스 기반이다.

embed 시스템은 부팅시간이 정말 짧다.

자동차도 os가 탑재되는데, 시동걸리는데 10초씩 걸리면 싫어할 것이다.

Q. fork하는 프로그램이 copy해오는 프로그램과 메모리 크기가 다를수도 있을것 같은데 그건 exec 시스템 콜에서 처리해주는 건가요?

A. 맞다. 정확하다.

Q. tv가 리눅스를 사용한다는 의미가 tv 본체를 말씀하시는건지, 올레티비 셋톱박스를 말씀하시는건지 궁금합니다

A. 대부분 요즘꺼는 리눅스를 쓴다. 옛날에는 자체 OS를 쓰기도 했었다.

Process creation in Linux

```
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <unistd.h>
int main(int argc, char* argv[]) {
  int counter = 0;
  pid_t pid;
  printf("Creating Child Process\n");
  pid = fork();  //이것이 핵심, fork를 하는 것이다.
  //pid 번호가 넘어오는 곳이 어디인가? pid=0이면 child 프로세스이다.
  //system call을 하는 경우에는 error 핸들링을 해줘야 하는 것이다. (이런 철학을 가져라)
  //1%의 exception을 처리해야 한다. error 핸들링을 missing하지 말라.

  if(pid < 0){ // Error in fork
    fprintf(stderr, "fork faild, errno: %d\n", errno);
    exit(EXIT_FAILURE);
  }

  else if(pid > 0){ // This is Parents Process
    int i;
    printf("Parents(%d) made Child(%d)\n", getpid(), pid);
    for(i=0; i<10; i++){
      printf("Counter: %d\n", counter++);
    }
  }

  else if(pid == 0){ // This is Child Process
    int i;
    printf("I am Child Process %d!\n", getpid());
    execl("/bin/ls", "ls", "-l", NULL); // Run 'ls -l' at /bin/ls
    //ls라는 프로그램을 실행시키는 것인데, shell은 이걸 실행시킬 수 있는 permission이 있다.
    for(i=0; i<10; i++){ // Cannot be run
      printf("Counter: %d\n", counter++);
    }
  }

  wait(NULL); //wait for child termination
  return EXIT_SUCCESS;
}
```

Q. 만약에 저 코드에서 pid값이 음수가 나온 경우의 에러를 처리해주지 않는다면 무슨 일이 생기나요?

A. 그거야 그 다음 코드가 어찌되느냐에 따라 다르다. 에러가 나면 child가 없다는 의미이다.

원래는 프로세스에 대해서 pid가 1대1 매칭이 되지만,

코드상에서의 0을 돌려주는 것은 제대로 만들어졌다는 것을 알려주기 위해서 0을 주는 것이다.

Q. fork 이후에 exec을 실행해줘야 하는 semantic을 개발자가 지켜야 한다면 하나의 system call 로 묶어서 제공하지 않는 이유가 있나요?

A. 리눅스커널을 제대로 만질줄 아는 사람들이 얼마나 많을 것 같은가?

매우 적다. 그리고 연봉이 높다.

이러한 코드들은 맥락이 있디. 시멘틱이 있다.

Process creation in Window

- CreateProcess Prototype

```
BOOL CreateProcess(
  LPCTSTR lpApplicationName,  // name of executable module
  LPTSTR lpCommandLine,     // command line string
  LPSECURITY_ATTRIBUTES lpProcessAttributes, // SD
  LPSECURITY_ATTRIBUTES lpThreadAttributes,  // SD
  BOOL bInheritHandles,                      // handle inheritance option
  DWORD dwCreationFlags,                     // creation flags
  LPVOID lpEnvironment,                      // new environment block
  LPCTSTR lpCurrentDirectory,                // current directory name
  LPSTARTUPINFO lpStartupInfo,               // startup information
  LPPROCESS_INFORMATION lpProcessInformation // process information
);
```

Process creation in Window (Cont’d)

```
/*CreateProcess.c*/
#include <stdio.h>
#include <windows.h>
enum{loop = 100};
int main(int argc, char *argv[]){
  STARTUPINFO si;
  PROCESS_INFORMATION pi;
  ZeroMemory(&pi, sizeof(PROCESS_INFORMATION));
  ZeroMemory(&si, sizeof(STARTUPINFO));
  si.cb = sizeof(si);
  si.lpReserved = NULL;
  si.lpTitle = L“ChildProcess.exe";
  si.lpDesktop = NULL;
  si.dwX = 0;
  printf("Parent Process ID: %d\n", GetCurrentProcessId());
  if(!CreateProcess( L"ChildProcess.exe", NULL, NULL, NULL, FALSE, 0, NULL, NULL, &si, &pi)){
    fprintf(stderr, "Create Process faild (%d)\n", GetLastError());
    return EXIT_FAILURE;
  }
  printf("Child Process ID: %d\n", pi.dwProcessId);
  {
    int i;
```

```
    for(i=0;i<loop;++i){
      printf("PID: %d, i: %d\n", GetCurrentProcessId(), i);
    }
  }
  WaitForSingleObject( pi.hProcess, INFINITE );
  printf("child process exit\n");
  CloseHandle( pi.hProcess );
  CloseHandle( pi.hThread );

  return EXIT_SUCCESS;
}
```

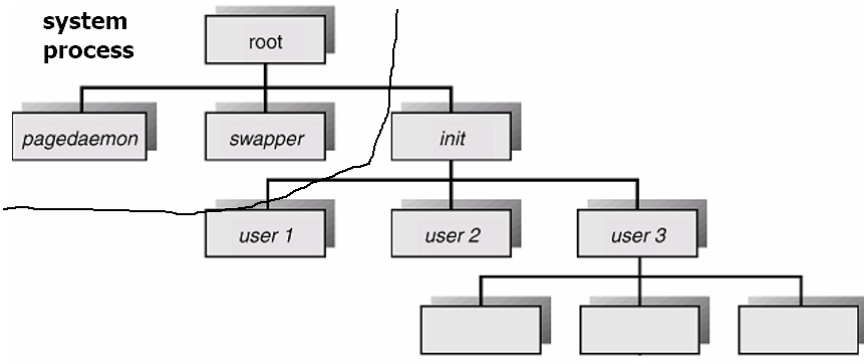
여기는 셋업해줘야 하는게 많다.

결국 Framework는 그 셋업에 대한 이해가 있어야 한다.

Process creation in Window (Cont'd)

```
/*ChildProcess.c*/
#include <stdio.h>
#include <windows.h>
enum{loop = 100};
 void ProcExit(void){
  printf("process exit\n");
}
int main(int argc, char *argv[]){
  int i; atexit(ProcExit);
  for(i=0;i<loop;++i){
    printf("PID: %d, i: %d\n", GetCurrentProcessId(), i);
  }
  return EXIT_SUCCESS;
}
```

Process Creation Hierarchy in UNIX



Process termination

RTS가 프로세스를 만들고 프로세스가 끝나면 RTS로 돌아간다.

그러면 RTS가 exit을 실행시켜준다.

그래야 이 프로그램을 만들때 쓰인 PCB같은 커널 data structure를 지울 수 있게 되는 것이다.

pid는 child를 만들때 마다 새로 생성이 된다.

다면 fork에서 return을 할때만 0을 돌려주는 것이다.

- Process termination
  - A process terminates when it finishes executing its final statement and asks kernel to delete it by using exit system call.
    - Output data from child to parent (via wait)
    - Process' resources are deallocated by kernel.
    - 만약 exit(2) 이 호출되지 않는다면

  - The abort function causes abnormal process termination.

    abort는 외부에서 비정상적으로 프로세스를 죽이는 것이다.
    - The SIGABRT signal is sent to the calling process.

      → SIGABRT이나 혹은 trap을 사용하여 abort하는 것이다.

      → trap도 사실 어디선가 abort를 불러와서 kenel data structure를 다 release한다.
    - core dump is made.

Cooperating Processes

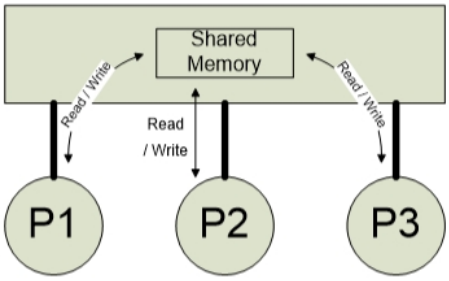
프로세스간의 협력이 필요한 경우가 생겼다.

협력, 그리고 상황에 따라 프로세스를 얼마나 만들 것인가에 대한 문제가 생겼다.

역할을 서로 나누고, shared memory를 통해 서로 data를 주고 받는 것이다.

- Independent process cannot affect or be affected by the execution of another process.
- Advantages of process cooperation.
  - Information sharing
  - Computation speed-up
  - Modularity
  - Convenience

(예) 빅데이터 처리 시스템



Q. 현재 실행되고 있는 여러 process들 중에 새롭게 실행될 process와 비슷한 memory space가 할당된 process가 있다면 그러한 process를 fork 하는 것이 성능상 좋을 것 같은데 그러한 선택의 과정도 이루어지나요?

A. 이렇게 구체적으로 판단을 안한다. 비슷한 정도가 얼마인가?

fork를 할 것인지의 그 판단은 간소하게 바이너리로 한다.

비슷한지의 여부를 따지는 것은 매우 어렵다.

요즘 메모리에서의 layout은 생각보다 굉장히 복잡하다.

Q. OS, RTS 모두 메모리에 올라와서 실행될텐데 프로세스라고 부르지 않는 이유가 있나요? 차이점은 무엇인지 궁금합니다.

A. OS와 RTS는 창조자이고, init같은 것은 창조물이다. 구분되어야 할 필요성이 있다.

kernel의 영역과 유저 스페이스가 구분되어야 할 필요성이 있다.

Q. Process termination에서 Output data from child to parent (via wait)가 위에서 설명했던 SIGCHLD를 설명하는건가요??

A. SIGCHLD은 signal이라는 것이다. Child가 Parent에게 뭔가를 알려주는데 쓰이는 메커니즘인 것이다. SIGCHLD은 child가 보내는 시그널이다.

Q. exit와 abort 둘 다 프로세스의 KDS를 해제해주는 동작을 하는것이 맞나요??

A. yes. 프로세스을 만들기 위해 커널의 쉐도우에서 커널의 data struture를 할당하고, 프로세스가 끝나면 해체해 줘야 한다.

Q. 프로그래밍 중 코드에서 쓰레드를 만드는 것이 프로세스에서 프로세스를 만드는 과정인 것인가요 아님 다른 것인가요?

A. 일단은 프로세스만 생각해봐라. 일단 두개의 과정은 다르다.

Q. 프로세스를 카피하는 이유가 각 프로세스가 소유하는 메모리에 공통으로 커널영역이 있고, 이를 효율적으로 메모리에 올리기 위해서인지 궁금합니다.

A. 정확하다. 나중에 메모리 management를 하는 것은 매우 복잡하다는 것을 알 것이다.

Q. parent에서 child 관계가 아닌 process를 만드는 것이 가능한가요? 종속적으로만 생성이 가능한 지 알고싶습니다.

A. NO. 불가능하다.

Q. child process가 parent process 보다 더 일찍 terminate 할 수도 있나요?

A. 당연하다. 독립적으로 수행되는 것이다. 아무 관계 없다.

Q. 어떤 process를 복사해도 상관없다면 init 은 첫 process를 만든 이후에는 필요가 없어지는건가요

A. init은 죽으면 안된다. init이 죽으면 새로운 프로세스를 만들 수가 없다.

→ init을 죽이는 방식의 해킹이 옛날에는 있었다.