Process Information

프로세스 정보

06강 요약

2016136092 이상민
2019136067 배수빈
2019136105 이지영

CONTENTS

01

프로세스

- Program
- Process
- ps, top, kill

02

프로세스 계층

- Process ID
- Process Group
- Session

03

프로세스 실행 시간

- Process Running Time
- System Running Time
- User Running Time

01

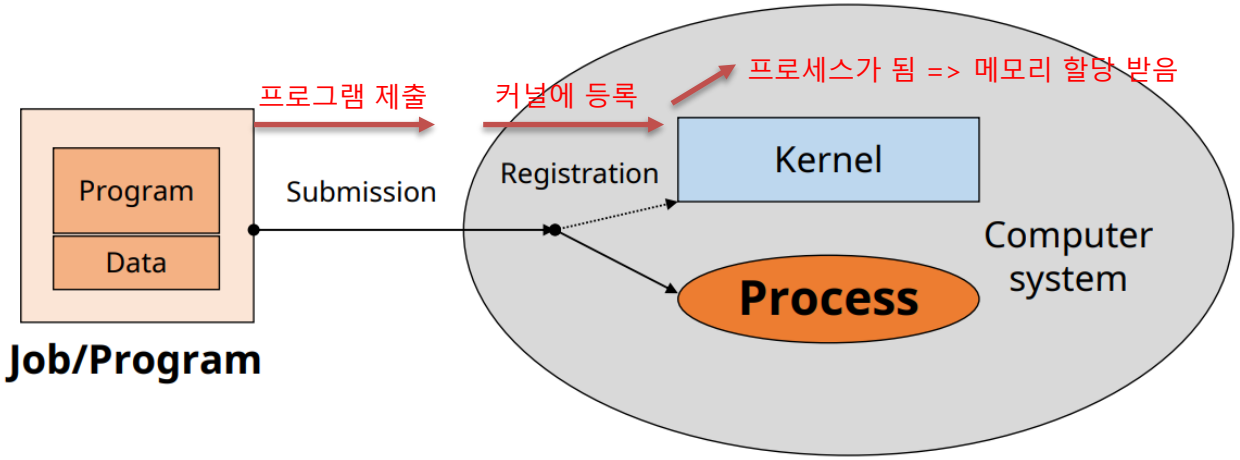
프로세스

프로세스(Process)

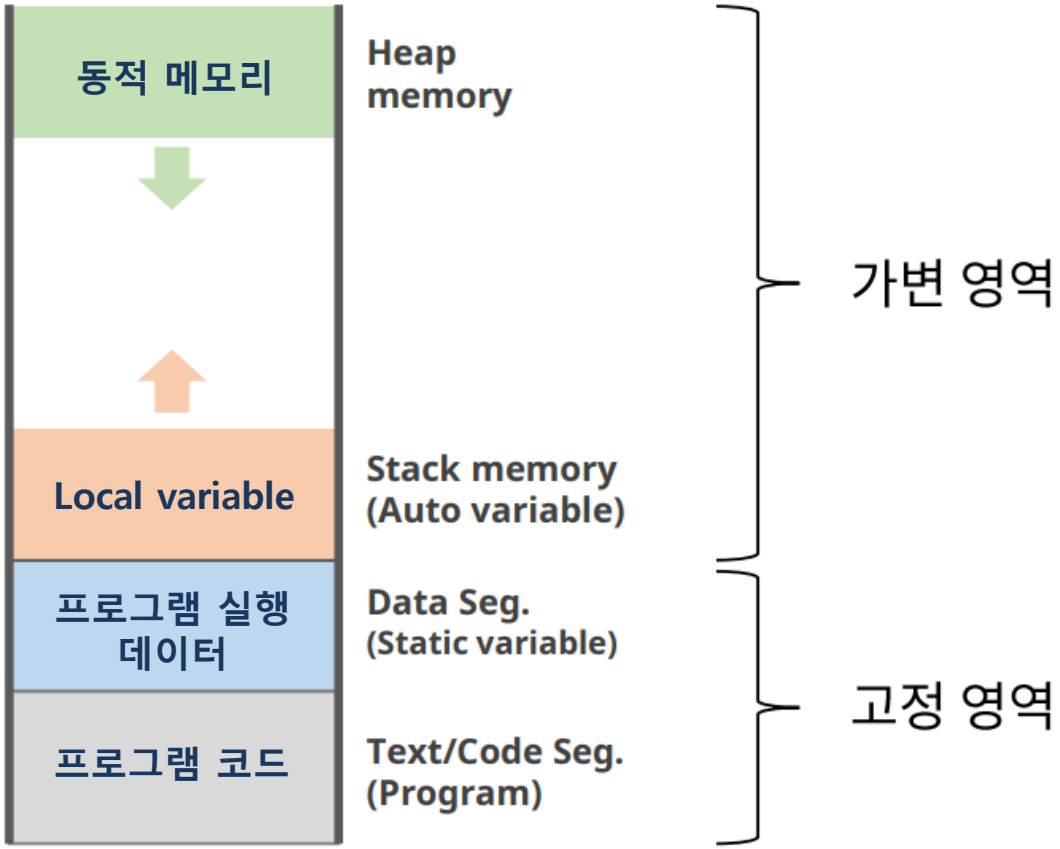
프로그램 (Program) VS 프로세스(Process)

- 실행 할 **프로그램 + 데이터(data)**
- 컴퓨터 시스템에 실행 요청 전의 상태

- 실행이 된(실행 중인) 프로그램 (Running Program)
- 실행을 위해 시스템(커널)에 등록된 작업
- 시스템 성능 향상을 위해 커널에 의해 관리 됨.



메모리 구조


동적 메모리
Heap memory
Local variable
Stack memory (Auto variable)
프로그램 실행 데이터
Data Seg. (Static variable)
프로그램 코드
Text/Code Seg. (Program)
가변 영역
고정 영역

Shell commands - Process

ps [options]

⇒ **현재 실행 중인 프로세스 정보 확인**

[options]

- -u userid : 특정 사용자의 프로세스 확인
- -j : job control format으로 출력(PID, PGID, SID 확인 가능)

top

⇒ **현재 시스템 내 프로세스 정보 실시간 확인**

- 윈도우 작업관리자 모드와 유사

kill [signal] <pid>

⇒ **프로세스에게 신호(signal) 전달**

[signal]

- -9 : 프로세스 강제 종료(SIGKILL)

02

프로세스 계층

Process IDs

Process ID

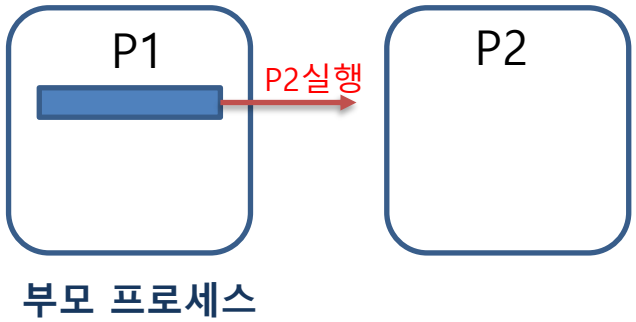
⇒ 프로세스에 부여된 식별 번호

- **PID**
- 시스템 내 유일한 번호로 할당 됨.

Parent Process ID

⇒ 자신을 생성한 프로세스에 부여된 식별 번호

- **PPID** : 부모 프로세스 ID
- 모든 프로세스는 부모프로세스가 있음
- 최상위 프로세스 : 커널 (pid = 0)



Process IDs

* User ID vs Process ID vs UIDs of a Process

User ID	Process ID	UIDs of a Process
UID	PID	RUID, EUID, SUID
사용자에게 부여된 식별 번호	프로세스에 부여된 식별 번호	RUID : 최초에 프로세스를 실행한 user의 ID
		EUID : 현재 프로세스가 행사하는 UID
		SUID : 프로세스의 최초의 EUID

System calls – Process ID

```
#include <unistd.h>
```

```
pid_t getpid (void);
```

⇒ **현재 프로세스 ID(PID) 반환**

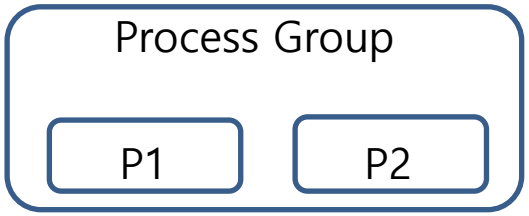
```
pid_t getppid (void);
```

⇒ **현재 프로세스의 부모 프로세스 ID(PPID) 반환**

Process Group

Process Group

⇒ 관련된 프로세스들을 묶은 것



- 하나의 job 수행을 목적으로 하는 프로세스
- 예) Shell script 내부 명령어들, 부모와 자식 프로세스 등
- Process Group에 전달된 **signal은 그룹 내 모든 프로세스에게 전달** 됨

- **Process Group Leader** : process group에 속한 프로세스 중 하나
- **Process Group ID (PGID)**
  - Process group에 부여된 고유 번호
  - PGID = Process Group Leader의 PID

System calls – Process Group ID

```
#include <unistd.h>
```

```
pid_t getpgid (pid_t pid);
pid_t getpgrd (void);
```

⇒ pid**의 Group ID 반환**

```
int setpgid (pid_t pid, pid_t pgid);
```

⇒ pid **프로세스를** pgid **프로세스 그룹으로 옮김**

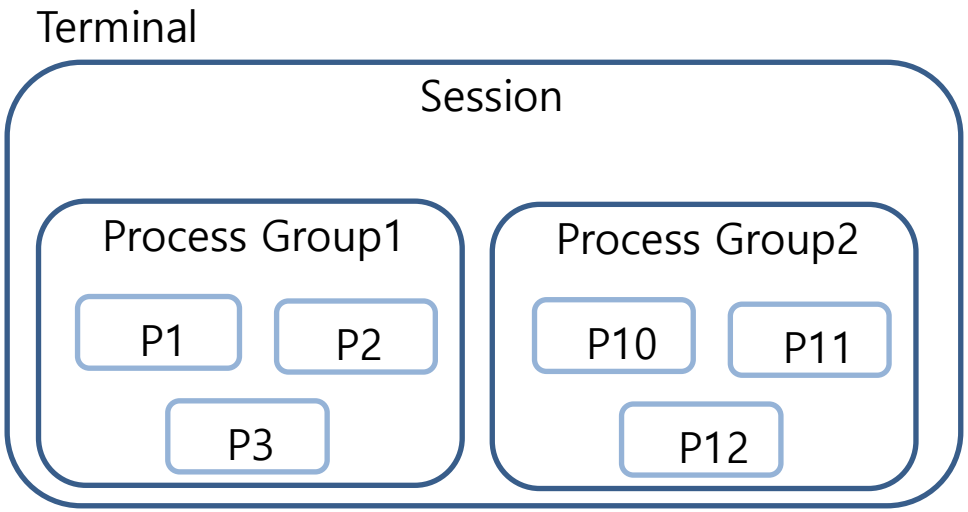
- **그룹 이동은 같은 Session내의 그룹만 가능**
- pid = 0인 경우, 현재 프로세스를 의미
- pgid = 0인 경우, pgid를 pid로 설정

Session

Session

⇒ 사용자가 로그인해 작업하고 있는 터미널 단위로 프로세스 그룹을 묶은 것

- 사용자 로그인 시, 새로운 session이 생성됨
  - 터미널 새로 실행 시
- Session 역할 : Process Group과 interrupt unit(e.g. 키보드 입력)을 연결하는 방법 등



Session

Session

- **Foreground Process Group**
  - ⇒ 사용자 IO를 수행하는 process group
  - 터미널을 통해 전달되는 signa을 전달받음
  - Session은 1개의 foreground process group을 가짐
- **Background Process Group**
  - ⇒ Foreground process group 외의 process group
  - Output은 터미널에 출력됨 (단, 입력은 불가)
  - Session은 0개 이상의 background process group을 가짐
- **Session Leader** : login shell process
- **Session ID** = Session Leader의 ID = SID

System calls – Session

```
#include <unistd.h>
```

```
pid_t getsid (pid_t pid);
```

⇒ **pid의 Session ID 반환**

```
int setsid (void);
```

⇒ **새로운 session 생성**

- setsid()를 호출한 프로세스가 현재 session leader가 아니면, 새로운 session을 생성하고 생성된 session의 리더가 됨
- 단, 생성된 session 은 controlling terminal을 갖지 않음

03

프로세스 실행 시간

프로세스 실행시간

Running Time of a Process

- 프로세스 실행시간 (Process running time)

  ⇒ **System running time + User running time**

- **System running time**

  ⇒ Kernel code를 수행한 시간 (System call에 소요된 시간)

- **User running time**

  ⇒ 사용자 모드에서 수행한 시간 (사용자 작성 코드를 실행하는데 걸린 시간)

- 실행시간 측정 및 사용 시간에 따른 요금 부과 등에 활용 가능

System calls – Process running time

```
#include <sys/times.h>
```

```
clock_t times (struct tms *buf);
```

⇒ **특정 시점부터 경과한 시간 반환 (Clock tick 단위)**

- **buf** : Running time 정보를 저장할 tms 구조체 포인터

tms 구조체

```
struct tms {
        clock_t tms_utime;          // user time, 사용자 시간
        clock_t tms_stime;          // system time, 시스템 시간
        clock_t tms_cutime;         // user time of children
        clock_t tms_cstime;         // system time of children
```

04

출처

이미지 & PPT 출처

- **이미지 :** 시스템 프로그래밍 강의자료(Lecture 6. Process information)
- **PPT :** http://minheeblog.tistory.com/category/PPT

THANK YOU

8팀