Chapter 9: Signal (3/5)

김선영
sunyzero@gmail(dot)com
버 전: 2017-05-27

가메 출판사 http://www.kame.co.kr
저자홈페이지 http://sunyzero.tistory.com

Session, Process Group

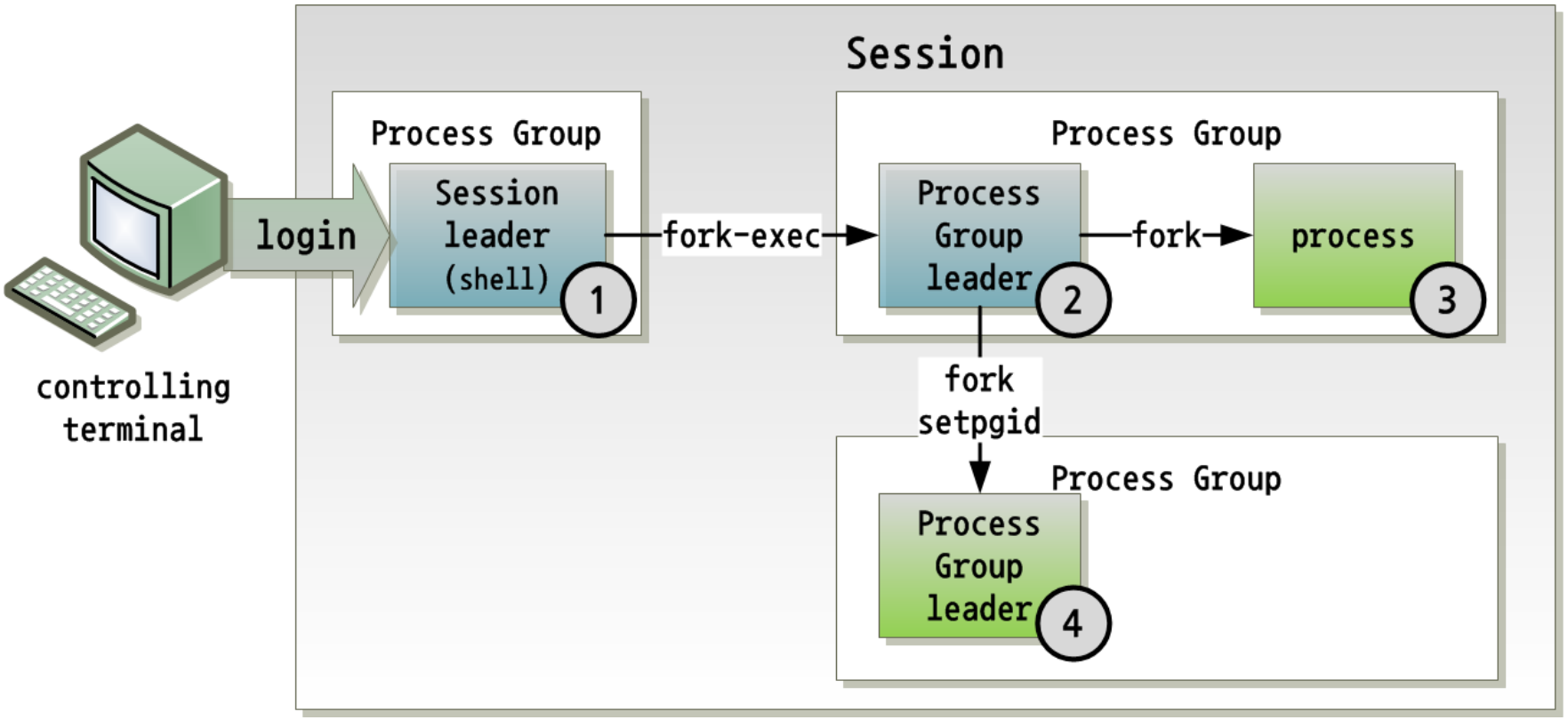
Session, ProsessGroup

- Session
  - Logical region (connection lifetime)
  - has several process groups
  - has a control terminal (optional)
- Process Group
  - Process group leader : ProcessGroup ID == PID
    - * 프로세스 그룹 리더는 시그널 전파 기능을 가진다.

Controlling terminal

- UNIX98 Pseudo terminals (PTS)
  - pty/# 형식 : pseudo terminal typewriter
- BSD style
  - ttyp## 형식

Session, ProcessGrp, Process



setpgid, setpgrp

```
int  setpgid(pid_t pid,  pid_t  pgid);
pid_t  getpgid(pid_t  pid);
int  setpgrp(void);
pid_t  getpgrp(void);
```

- Process group
  - setpgid, getpgid : X/Open
  - setpgrp, getpgrp : BSD
- e.g.
  - setpgid(0, 0) = setpgid(getpid(), getpid()) = Process group leader

sig_pgid.c (1/3)

```
void sa_handler_usr(int signum);
int  main() {
    int     i;
    struct sigaction sa_usr1, sa_usr2;

    memset(&sa_usr1, 0, sizeof(struct sigaction));
    sa_usr1.sa_handler = sa_handler_usr;
    sigfillset(&sa_usr1.sa_mask);
    memset(&sa_usr2, 0, sizeof(struct sigaction));
    sa_usr2.sa_handler = sa_handler_usr;
    sigfillset(&sa_usr2.sa_mask);

    sigaction(SIGUSR1, &sa_usr1, NULL);
    sigaction(SIGUSR2, &sa_usr2, NULL);
    printf("++ PID(%d) PGID(%d) SID(%d)\n",
        getpid(), getpgid(0), getsid(0));
```

sig_pgid.c (2/3)

```
    for(i=0; i<3; i++) {
        if (fork() == 0) break;
    }
    for(;;) {
        pause();
        printf("PID(%d) Recv SIGNAL...\n", getpid());
    }
    return EXIT_SUCCESS;
}
```

sig_pgid.c (3/3)

```
void  sa_handler_usr(int signum) {
    switch(signum) {
        case SIGUSR1:
            printf("-- PID(%d) PGID(%d) SID(%d)\n",
                getpid(), getpgid(0), getsid(0));
            break;
        case SIGUSR2:
            if (getpid() != getpgid(0)) {
                setpgid(0, 0); /* 새로운 프로세스 그룹 생성 */
            } else {
                /* 부모 프로세스 그룹으로 변경(돌아온 탕자 프로세스) */
                setpgid(0, getppid());
            }
            printf("-- PID(%d) to PGID(%d)\n", getpid(), getpgid(0));
            break;
    }
}
```

sig_pgid : example

```
$ ./sig_pgid
++ PID(4285) PGID(4285) SID(3347)
-- PID(4288) to PGID(4288)

PID(4288) Recv SIGNAL...

$
```

Ctrl-C

```
$ ps -ejf ¦ grep sig_pgid
UID        PID  PPID  PGID   SID  C STIME TTY          TIME CMD
sunyzero  4285  3347  4285  3347  0 13:06 pts/2    00:00:00 ./sig_pgid
sunyzero  4286  4285  4285  3347  0 13:06 pts/2    00:00:00 ./sig_pgid
sunyzero  4287  4285  4285  3347  0 13:06 pts/2    00:00:00 ./sig_pgid
sunyzero  4288  4285  4285  3347  0 13:06 pts/2    00:00:00 ./sig_pgid

$ kill -USR2 4288

$ ps -ejf ¦ grep sig_pgid
UID        PID  PPID  PGID   SID  C STIME TTY          TIME CMD
sunyzero  4285  3347  4285  3347  0 13:06 pts/2    00:00:00 ./sig_pgid
sunyzero  4286  4285  4285  3347  0 13:06 pts/2    00:00:00 ./sig_pgid
sunyzero  4287  4285  4285  3347  0 13:06 pts/2    00:00:00 ./sig_pgid
sunyzero  4288  4285  4288  3347  0 13:06 pts/2    00:00:00 ./sig_pgid

$ ps -ejf ¦ grep sig_pgid
sunyzero  4288     1  4288  3347  0 13:06 pts/2    00:00:00 ./sig_pgid
```

setsid

```
pid_t  getsid(pid_t  pid);
pid_t  setsid(void);
```

- session leader
  - SID == PID 인 프로세스
  - 당연히 process group leader가 된다.

Tip: daemon

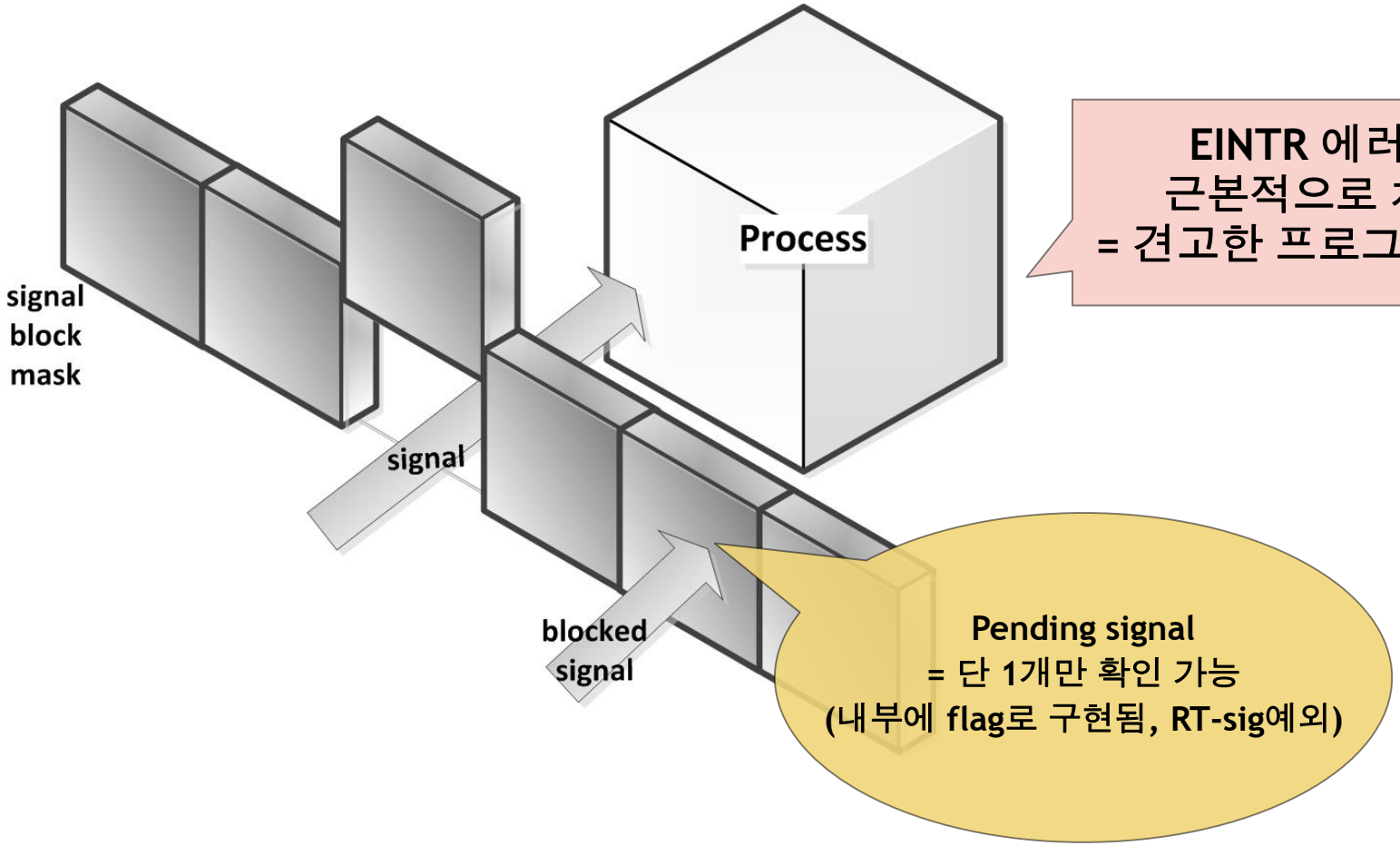
1. fork 성공 후 orphan process로 만든다.
   - init가 step parent가 된다.
2. setsid : session leader (no-conrol-terminal)
3. chdir("/")
4. umask(0)
5. close all opened files : include fd(0, 1, 2)

Signal to pgrp

- kill 명령시
  - 음수 PID = PGID의미
- kill(2)
  - kill(-4285, SIGTERM)
  - = killpg(4285, SIGTERM)

Signal block mask

Signal block mask



Sig block mask : Pending signal

- Avoid unpredictable event
  - I/O processing
  - Critical section


```
int sigprocmask(int how, const sigset_t *set, sigset_t *oldset);
int sigpending(sigset_t *set);
int sigsuspend(const sigset_t *mask);  /* pause() + signal mask */
```

how	Description
SIG_BLOCK	지정된 **set**을 **block mask**에 추가
SIG_UNBLOCK	지정된 **set**을 **block mask**에서 제거
SIG_SETMASK	지정된 **set**으로 **block mask**를 설정

sig_pending.c (1/3)

```
void  sa_handler_usr(int signum);  /* SIGU
int  main() {
    int     i;
    struct sigaction   sa_usr1, sa_usr2;
    sigset_t        sigset_mask, sigset_oldmask, sigset_pend;

    memset(&sa_usr1, 0, sizeof(struct sigaction));
    sa_usr1.sa_handler = sa_handler_usr;
    sigfillset(&sa_usr1.sa_mask);
    sigaction(SIGUSR1, &sa_usr1, NULL);
    sa_usr2.sa_handler = SIG_IGN;    /* USR2 핸들러는 무시로 변경한다. */
    sigaction(SIGUSR2, &sa_usr2, NULL);

    sigfillset(&sigset_mask);        /* 모든 시그널 매스크를 채움 */
    sigdelset(&sigset_mask, SIGINT); /* 시그널 매스크에서 SIGINT 삭제 */
    printf("PID(%d)\n", getpid());
```

sig_pending.c (2/3)

```
 for(;;) {
        printf("Install signal block mask (allow only SIGINT)\n");
        sigprocmask(SIG_SETMASK, &sigset_mask, &sigset_oldmask);
        sleep(10);
        sigpending(&sigset_pend); /* check blocked signal */
        for (i=1; i<SIGRTMIN; i++) {
            if (sigismember(&sigset_pend, i)) {
                printf("\tPending signal = %d\n", i);
                switch(i) {
                    case SIGUSR2:
                        sa_handler_usr(SIGUSR2);  /* 수동으로 지연된 시그널 처리 */
                        break;
                    default:
                        break;
                } /* end: switch */
            } /* end: if */
        } /* loop: for(i) */
        printf("Restore the previous signal block mask.\n");
        sigprocmask(SIG_SETMASK, &sigset_oldmask, NULL); /* sigblockmask 복구 */
    } /* loop: for */
    return 0;
}
```

sig_pending.c (3/3)

```
void  sa_handler_usr(int signum) {
    int i;
    for (i=0; i<3; i++) {
        printf("\tSignal(%s):%d sec.\n",
            signum == SIGUSR1 ? "USR1":"USR2", i);
        sleep(1);
    }
}
```

시그널 매스크를 이용해서 지연된 시그널을 처리하는 방법은
중요한 I/O처리나 critical section처리에 꼭 사용되는 기법이다.
단 SIGTERM이나 SIGABRT는 블록하면 안된다. 비정상 동작때 죽이거나
core를 덤프할 수 있도록 해야 하기 때문이다.

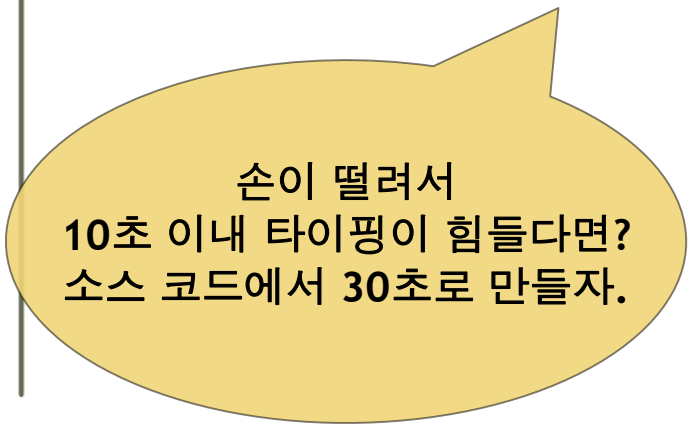
sig_pending : example

```
$ ./sig_pending
PID(1794)
Install signal block mask (allow only SIGINT)
        Pending signal = 10
        Pending signal = 12
        Signal(USR2):0 sec.
        Signal(USR2):1 sec.
        Signal(USR2):2 sec.
Restore the previous signal block mask.
        Signal(USR1):0 sec.
        Signal(USR1):1 sec.
        Signal(USR1):2 sec.
Install signal block mask (allow only SIGINT)
```

```
$ kill -USR1 1794

$ kill -USR2 1794
```

10초이내에 타이핑



kill -USR1 1794 대신에 kill -USR1 $(pgrep -f ./sig_pending)으로 명령을 내려도 된다.

volatile vs signal

- sig_atomic_t : 정수형

```
volatile sig_atomic_t   gi_counter;
```

  - 최적화 금지를 위해 volatile과 같이 사용
    - SUSv2부터 sig_atomic_t는 volatile을 포함
    - volatile은 2번 쓰여도 문법 오류가 아님!
  - 대부분의 implementation에서는 32bit를 커버
  - volatile은 휘발성이 있는 공간 : no-caching