운영체제론 실습

- 프로세스 목록 나타내는 모듈 프로그래밍

목 차

- **Task struct**
  - 프로세스 정보를 나타내는 구조체 설명
- **프로세스 목록 출력 프로젝트 설명**
  - 프로젝트 설명
  - 프로젝트에 필요한 함수들(....)
  - 프로젝트 결과화면

예제 코드 다운로드 경로

아래 명령어를 linux 환경에서 치면 다운받을 수 있음.

$ wget http://ce.hanyang.ac.kr/week5.tar

ps 명령어

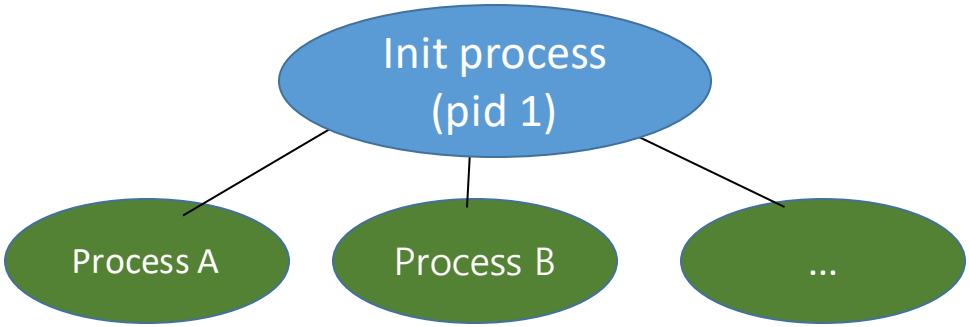
- ps : 실행중인 프로세스들의 정보를 출력하는 명령어

```
os@os:~$ ps aux
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  1.4  0.0 225412  9024 ?        Ss   10:40   0:04 /sbin/init splash
root           2  0.0  0.0      0     0 ?        S    10:40   0:00 [kthreadd]
root           3  0.0  0.0      0     0 ?        I<   10:40   0:00 [rcu_gp]
root           4  0.0  0.0      0     0 ?        I<   10:40   0:00 [rcu_par_gp]
root           5  0.0  0.0      0     0 ?        I    10:40   0:00 [kworker/0:0-eve]
root           6  0.0  0.0      0     0 ?        I<   10:40   0:00 [kworker/0:0H-kb]
root           7  0.0  0.0      0     0 ?        I    10:40   0:00 [kworker/u8:0-ev]
root           8  0.0  0.0      0     0 ?        I<   10:40   0:00 [mm_percpu_wq]
root           9  0.0  0.0      0     0 ?        S    10:40   0:00 [ksoftirqd/0]
root          10  0.0  0.0      0     0 ?        I    10:40   0:00 [rcu_sched]
root          11  0.0  0.0      0     0 ?        S    10:40   0:00 [migration/0]
root          12  0.0  0.0      0     0 ?        I    10:40   0:00 [kworker/0:1-eve]
root          13  0.0  0.0      0     0 ?        S    10:40   0:00 [cpuhp/0]
root          14  0.0  0.0      0     0 ?        S    10:40   0:00 [cpuhp/1]
root          15  0.0  0.0      0     0 ?        S    10:40   0:00 [migration/1]
root          16  0.0  0.0      0     0 ?        S    10:40   0:00 [ksoftirqd/1]
root          18  0.0  0.0      0     0 ?        I<   10:40   0:00 [kworker/1:0H-kb]
root          19  0.0  0.0      0     0 ?        S    10:40   0:00 [cpuhp/2]
root          20  0.0  0.0      0     0 ?        S    10:40   0:00 [migration/2]
root          21  0.0  0.0      0     0 ?        S    10:40   0:00 [ksoftirqd/2]
root          22  0.0  0.0      0     0 ?        I    10:40   0:00 [kworker/2:0-pm]
root          23  0.0  0.0      0     0 ?        I<   10:40   0:00 [kworker/2:0H-kb]
root          24  0.0  0.0      0     0 ?        S    10:40   0:00 [cpuhp/3]
root          25  0.0  0.0      0     0 ?        S    10:40   0:00 [migration/3]
root          26  0.0  0.0      0     0 ?        S    10:40   0:00 [ksoftirqd/3]
root          27  0.0  0.0      0     0 ?        I    10:40   0:00 [kworker/3:0-cgr]
root          28  0.0  0.0      0     0 ?        I<   10:40   0:00 [kworker/3:0H-kb]
root          29  0.0  0.0      0     0 ?        S    10:40   0:00 [kdevtmpfs]
root          30  0.0  0.0      0     0 ?        I<   10:40   0:00 [netns]
```

Task struct

task_struct

- 프로세스가 생기면서 이를 관리하기 위해 같이 만들어지는 구조체
- 프로세스에 관한 모든 정보를 보관하는 프로세스 서술자
  - 사용중인 파일, 프로세스의 주소 공간, 프로세스 상태 등
- Linux가 fork를 통해서 모든 프로세스를 생성하기 때문에 가장 처음 생기는 프로세스인 init process에서 모든 프로세스들이 만들어진다.



- init process와 processA는 부모-자식 관계
- process A와 process B는 sibling관계

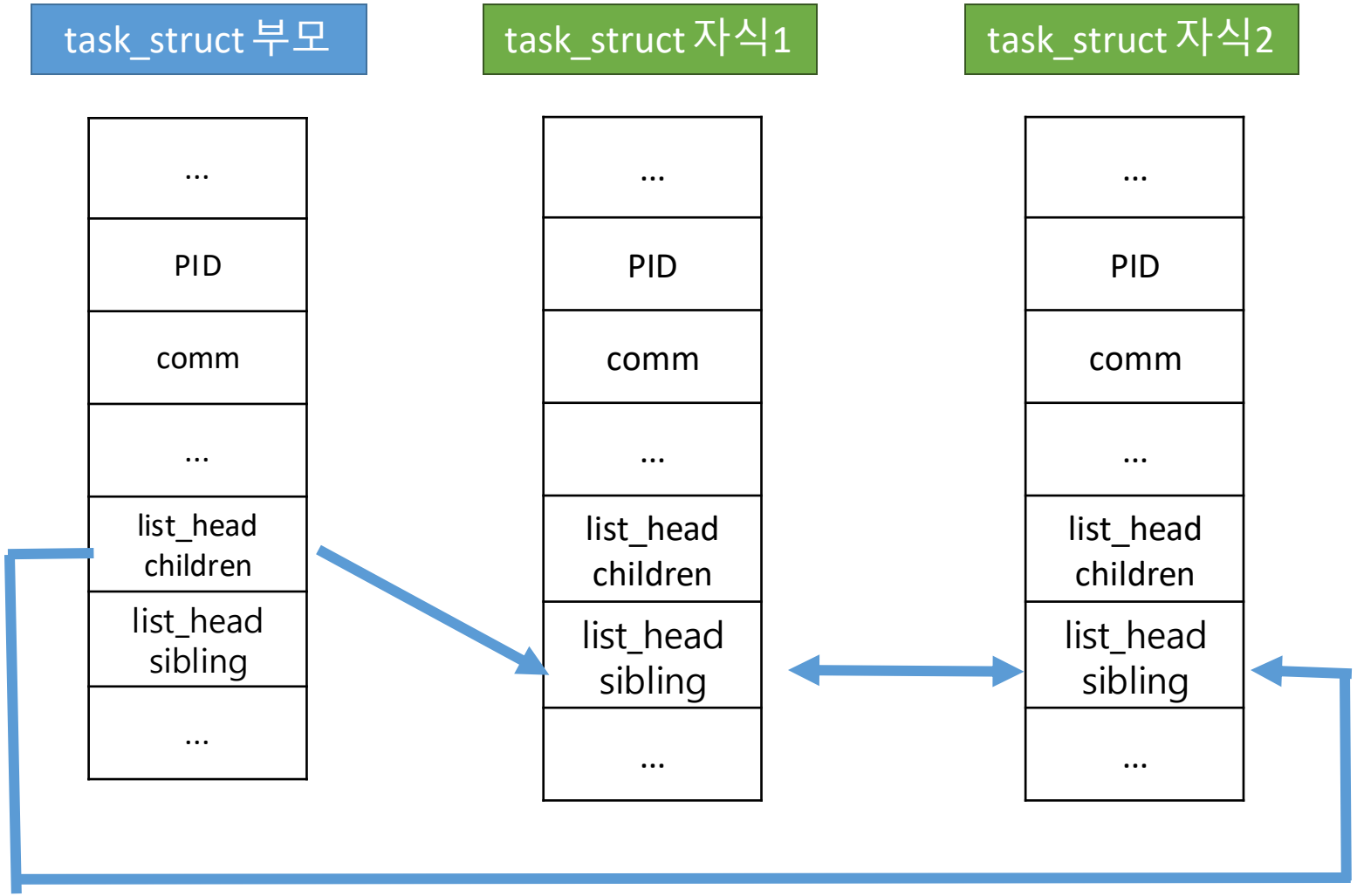
task_struct

- https://elixir.bootlin.com/linux/latest/source/include/linux/sched.h#L592

본 실습에 필요한 속성만 보기 :

```
struct task_struct {

    /* process id */
1   pid_t       pid;

    /* executable name, excluding path. */
2   char        comm[TASK_COMM_LEN];

    /* -1 unrunnable, 0 runnable, >0 stopped: */
3   volatile long     state;

    /* Children/sibling form the list of natural children */
4   struct list_head    children;
    struct list_head    sibling;
    struct task_struct    *group_leader;

}
```

task_struct 파악하기



for_each_process

- https://elixir.bootlin.com/linux/latest/source/include/linux/sched/signal.h#L542

/ include / linux / sched / signal.h

```
542  #define for_each_process(p) \
543          for (p = &init_task ; (p = next_task(p)) != &init_task ; )
```

- init process를 시작으로 프로세스 리스트를 탐색

예제1: for_each_process

- for_each_process 매크로를 사용해
  init process부터 실행된 모든 프로세스들의 command와 pid를 출력

File Edit View Search Terminal Help

```
 1 #include <linux/kernel.h>
 2 #include <linux/module.h>
 3 #include <linux/init.h>
 4 #include <linux/sched.h>
 5 #include <linux/sched/signal.h>
 6
 7 int module_start(void) {
 8     struct task_struct *task;
 9
10     printk(KERN_INFO "Init Module....");
11     for_each_process(task) {
12         printk("%s[%d]\n", task->comm, task->pid);
13     }
14     return 0;
15 }
16
17 void module_end(void) {
18     printk("Module removing...");
19 }
20
21 module_init(module_start);
22 module_exit(module_end);
~
~
~
~
~
~
~
                                                    13,2-5        All
```

예제1: for_each_proces 실행결과

```
os@os:~/os2019/week6/all_process$ sudo insmod all_pid_with_mecro.ko
[sudo] password for os:
os@os:~/os2019/week6/all_process$ dmesg
```

```
ernel.
[  498.174172] Disabling lock debugging due to kernel taint
[  498.174195] all_pid_with_mecro: module verification failed: signature
 and/or required key missing - tainting kernel
[  498.174339] Init Module....
[  498.174340] systemd[1]
[  498.174341] kthreadd[2]
[  498.174342] rcu_gp[3]
[  498.174343] rcu_par_gp[4]
[  498.174343] kworker/0:0H[6]
[  498.174344] kworker/u8:0[7]
[  498.174345] mm_percpu_wq[8]
[  498.174345] ksoftirqd/0[9]
[  498.174346] rcu_sched[10]
[  498.174347] migration/0[11]
[  498.174348] cpuhp/0[13]
[  498.174348] cpuhp/1[14]
[  498.174349] migration/1[15]
[  498.174350] ksoftirqd/1[16]
[  498.174350] kworker/1:0[17]
[  498.174351] kworker/1:0H[18]
[  498.174352] cpuhp/2[19]
[  498.174352] migration/2[20]
[  498.174353] ksoftirqd/2[21]
```

예제2: process의 부모-자식 관계 파악해보기

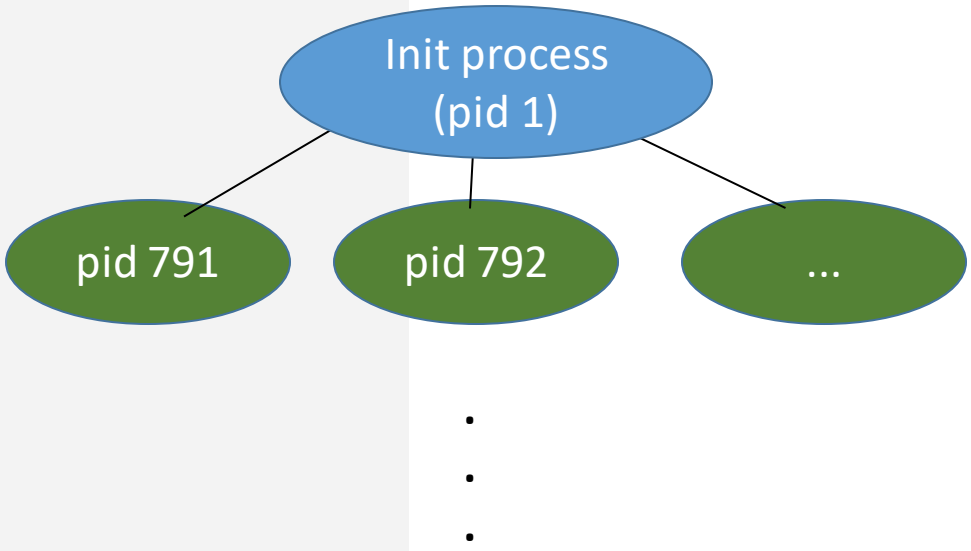
- 어떤 프로세스가 어떤 자식 프로세스를 생성했는지 출력
  (부모 및 자식 프로세스의 pid, command 출력)

```
 1 #include <linux/kernel.h>
 2 #include <linux/module.h>
 3 #include <linux/init.h>
 4 #include <linux/sched.h>
 5 #include <linux/sched/signal.h>
 6
 7 int module_start(void) {
 8     struct task_struct *task;
 9     struct task_struct *child;
10     struct list_head *list;
11
12     printk("MODULE INSTALLING...");
13     for_each_process(task) {
14         printk("\n %4d task %s\n children: ", task->pid, task->comm);
15         list_for_each(list, &task->children) {
16             child = list_entry(list, struct task_struct, sibling);
17             printk(" %4d %s", child->pid, child->comm);
18         }
19     }
20     return 0;
21 }
22
23 void module_end(void) {
24     printk("MODULE REMOVING...");
25 }
26
27 module_init(module_start);
28 module_exit(module_end);
~
~
"parent_child.c" 28L, 640C written                          28,5          All
```

예제2: process의 부모-자식 관계 파악해보기

- 어떤 프로세스에 의해 어떤 자식 프로세스가 생성되었는지 확인
- e.g) systemd에 의해 init process (pid 1)이 실행되었고 pid 791, 792, ... 등 많은 프로세스들이 init process의 자식 프로세스로 생성됨을 확인

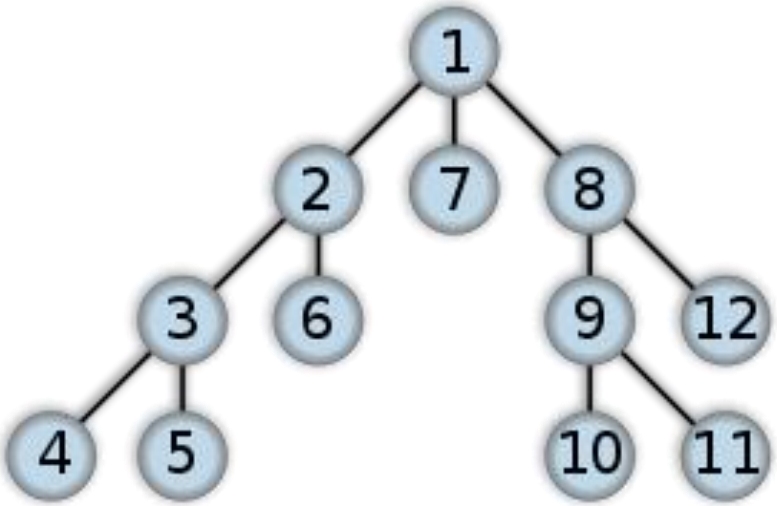
```
File  Edit  View  Search  Terminal  Help
[ 5256.975348] MODULE INSTALLING...
[ 5256.975355]
                    1 task systemd
                  children:
[ 5256.975359]      791 bluetoothd
[ 5256.975363]      792 thermald
[ 5256.975366]      793 udisksd
[ 5256.975369]      795 irqbalance
[ 5256.975372]      796 snapd
[ 5256.975375]      798 avahi-daemon
[ 5256.975378]      800 ModemManager
[ 5256.975381]      801 rsyslogd
[ 5256.975384]      804 acpid
[ 5256.975387]      805 networkd-dispat
[ 5256.975390]      806 systemd-logind
[ 5256.975393]      808 cron
[ 5256.975397]      819 accounts-daemon
[ 5256.975400]      820 dbus-daemon
[ 5256.975403]      858 NetworkManager
[ 5256.975406]      924 polkitd
[ 5256.975409]      945 unattended-upgr
[ 5256.975412]      989 gdm3
[ 5256.975415]     1108 systemd
[ 5256.975418]     1152 whoopsie
[ 5256.975422]     1159 kerneloops
[ 5256.975425]     1164 upowerd
[ 5256.975428]     1168 kerneloops
[ 5256.975431]     1288 rtkit-daemon
[ 5256.975434]     1502 ibus-x11
[ 5256.975437]     1568 boltd
```



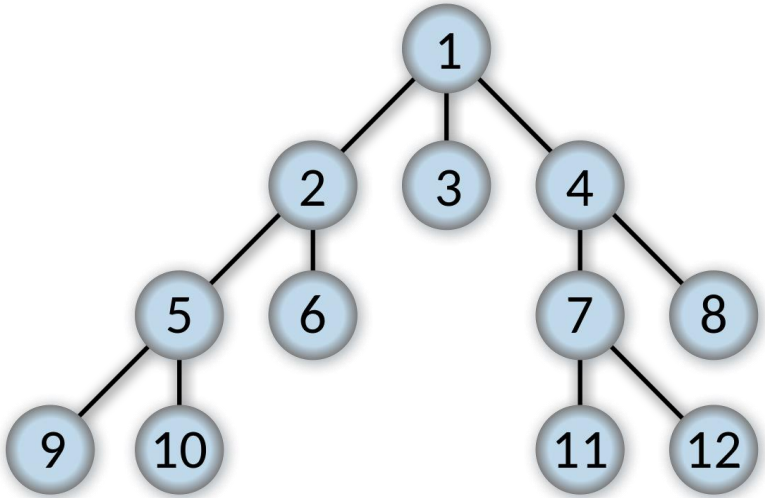
Depth-First Search

- 깊이를 우선 탐색하는 알고리즘
- 비교 대상으로는 너비 우선 탐색이 있음

Depth-First Search



Breadth-First Search



Depth-First Search

- **PSEUDO CODE**

```
dfs(i) {
        print information of i
        for each s ∈ siblings of child of i
            dfs(s);
}
init() {
        initial  task t
        dfs(t)
}
```

- **구현 시 필요한 함수와 속성**

- **list_for_each_entry**
- **task_struct -> children**
- **task_struct -> sibling**

 # PSEUDO CODE는 PSEUDO CODE 일 뿐, 각 함수에서 요구하는 타입은 이해를 바탕으로 구현해야 함

프로젝트:
프로세스의 pid 데이터 출력해주는 모듈

프로젝트: ps lite

- **목표: 기존 ps 명령의 lite 버전 구현**
  - init process부터 모든 프로세스의 pid 데이터를 출력해주는 모듈

수고하셨습니다.

- 다음 시간: **Sudoku 답 검증기**