

운영체제론 실습 6주차

정보보호연구실 @ 한양대학교

운영체제론 실습 6주차

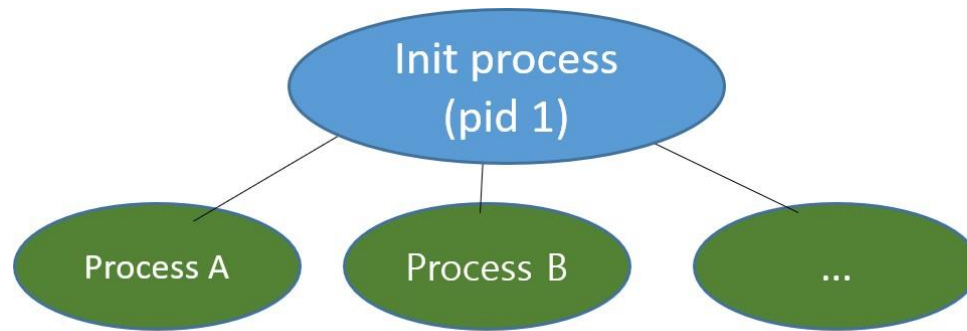
1. Task_struct

2. 프로세스 목록 출력 모듈 프로그래밍 (list_tasks_dfs)

Task_struct(PCB)

1) Task_struct(PCB)

- 프로세스가 생기면서 이를 관리하기 위해 만들어지는 구조체
- 프로세스에 관한 모든 정보를 보관하는 프로세스 서술자
 - 사용중인 파일, 프로세스의 주소 공간, 프로세스 상태 등
- Linux는 fork를 통해서 모든 프로세스를 생성
 - 최초 init 프로세스로부터 모든 프로세스들이 만들어짐



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

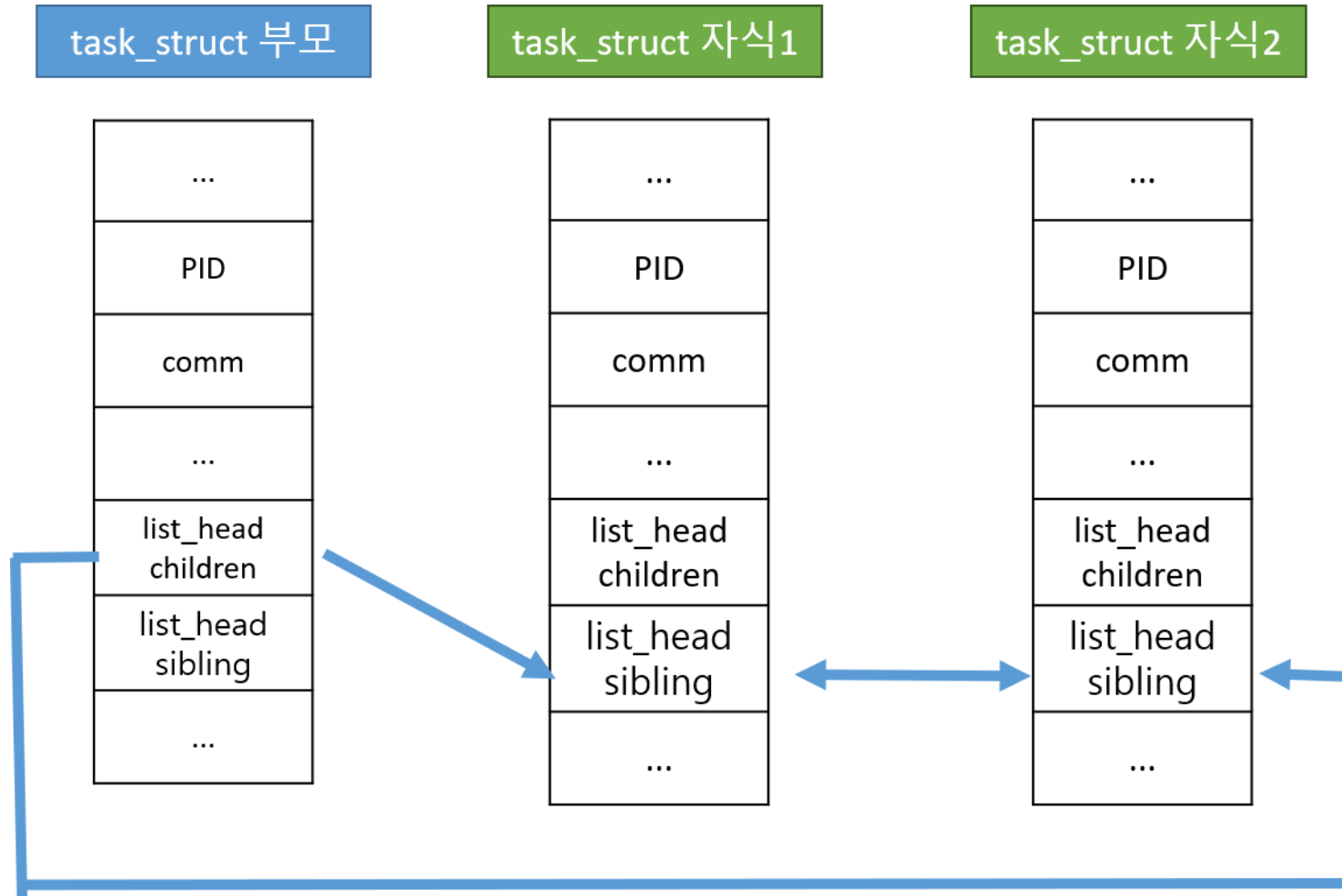
Task_struct

<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  
541  
542 #define for_each_process(p) \  
543     for (p = &init_task ; (p = next_task(p)) != &init_task ; )
```

- Init process를 시작으로 프로세스 리스트를 탐색

예제 1: for_each_process

- for_each_process를 사용해 init process부터 실행된 모든 프로세스들의 command와 pid 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);
~
~
```

예제 1: for_each_process 실행결과

- for_each_process를 사용해 init process부터 실행된 모든 프로세스들의 command와 pid출력

```
dongmin1@dongmin1-VirtualBox:~/week6/1_for_each_process$ sudo insmod all_pid_with_macro.ko  
[sudo] password for dongmin1:  
dongmin1@dongmin1-VirtualBox:~/week6/1_for_each_process$ dmesg
```

```
[24538.466767] Disabling lock debugging due to kernel taint  
[24538.469955] Init Module....  
[24538.469956] systemd[1]  
[24538.469957] kthreadd[2]  
[24538.469959] rcu_gp[3]  
[24538.469960] rcu_par_gp[4]  
[24538.469961] kworker/0:0H[6]  
[24538.469961] mm_percpu_wq[9]  
[24538.469962] ksoftirqd/0[10]  
[24538.469963] rcu_sched[11]  
[24538.469964] migration/0[12]  
[24538.469965] idle_inject/0[13]  
[24538.469966] cpuhp/0[14]  
[24538.469967] kdevtmpfs[15]  
[24538.469968] netns[16]  
[24538.469969] rcu_tasks_kthre[17]  
[24538.469970] kauditd[18]  
[24538.469971] khungtaskd[19]  
[24538.469972] oom_reaper[20]  
[24538.469973] writeback[21]
```


예제2: process의 부모-자식 관계 추적

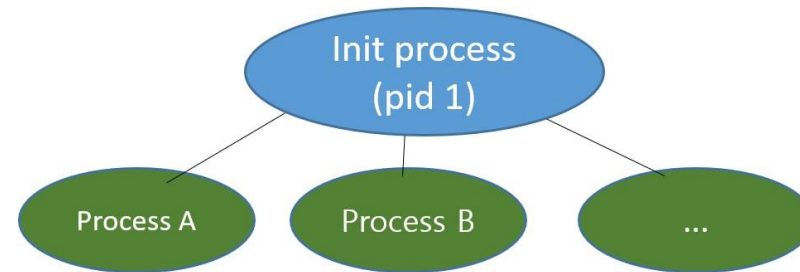
- 어떤 프로세스가 어떤 자식 프로세스를 생성했는지 출력
 - (부모 및 자식 프로세스의 pid, command 출력)

```
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 추적

- 어떤 프로세스에 의해 어떤 자식 프로세스가 생성되었는지 확인
- Systemd에 의해 init process (pid 1)이 실행되었고, pid 286, 302, ... 등 많은 프로세스들이 init process의 자식 프로세스로 생성됨을 확인

```
[27920.958165] INSTALL: parent_child
[27920.958176]
                1 task systemd
                children:
[27920.958178]    286 systemd-journal
[27920.958179]    302 systemd-udev
[27920.958180]    409 systemd-resolve
[27920.958181]    565 acpid
[27920.958182]    569 ModemManager
[27920.958183]    574 snapd
[27920.958184]    575 accounts-daemon
[27920.958194]    578 avahi-daemon
[27920.958195]    579 rsyslogd
[27920.958196]    598 cron
[27920.958197]    601 dbus-daemon
[27920.958198]    672 wpa_supplicant
[27920.958199]    673 NetworkManager
[27920.958200]    678 systemd-logind
[27920.958201]    679 networkd-dispatcher
[27920.958202]    681 udisksd
[27920.958203]    692 polkitd
```

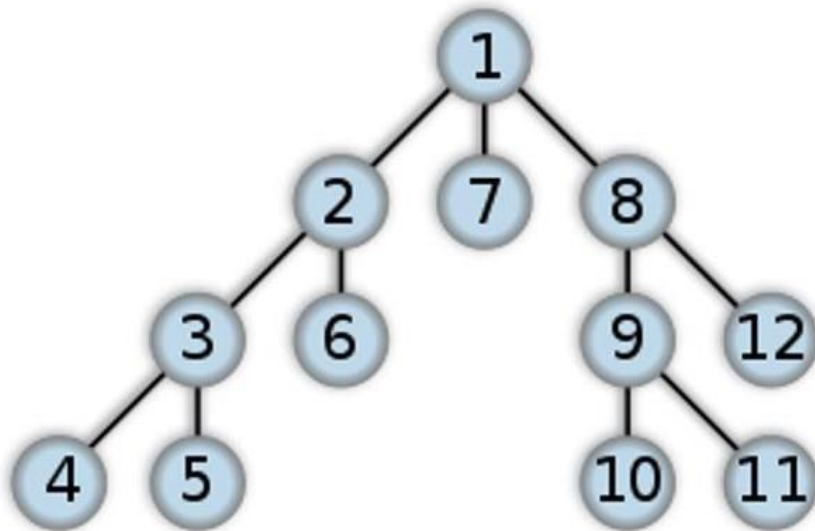


- init process와 processA는 부모-자식 관계
- process A와 process B는 sibling관계

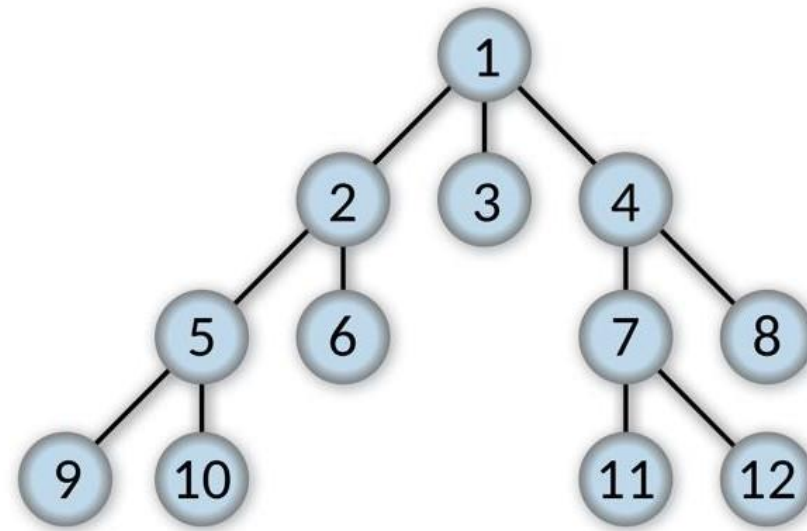
Depth-First Search

- 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)  
}
```

- 구현시 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

실습: ps lite

예제코드 경로:
week6/4_project/list_tasks_dfs.c

• 목표: 기존 ps명령의 lite 버전 구현

init process부터 모든 프로세스의 pid 데이터를 출력해주는 모듈

• Dfs 함수 구현

- [list_for_each] OR [list_for_each_entry] 둘 중 하나를 이용해 탐색
- list_entry(ptr, type, member) : 구조체 멤버의 포인터를 이용해 구조체의 시작주소를 찾는 역할, 이 매크로를 이용해 next_task의 주소값 갱신
- list_for_each_entry: 14p 참조

• Init 함수 구현

- find_get_pid(x): x pid의 구조체를 가져온다.
- pid_task(init_pid, *init_task): 가져온 pid를 통해 해당 프로세스의 task_struct 구조체를 가져온다.

list_for_each_entry

- list_for_each_entry 매크로 함수를 사용
 - 반복적으로 탐색하며 노드를 확인

```
/**
 * list_for_each_entry - iterate over list of given type
 * @pos:      the type * to use as a loop cursor.
 * @head:     the head for your list.
 * @member:   the name of the list_head within the struct.
 */
#define list_for_each_entry(pos, head, member) \
    for (pos = list_first_entry(head, typeof(*pos), member); \
         &pos->member != (head); \
         pos = list_next_entry(pos, member))
```

실습: ps lite

예제 코드 경로:
week6/4_project/list_
tasks_dfs.c

• 목표: 기존 ps명령의 lite 버전 구현

- 수행중인 프로세스의 조상을 찾아가는 방법
 - Sched.h로 부터 변수 current를 가져올 수 있다. 이는 현재 실행중인 프로세스를 의미한다.
 - 반복문을 통해 부모프로세스를 찾아간다. (init프로세스의 pid는 1)

예제)

```
init_task = current;  
while (init_task->pid != 1) { // pid가 1이 될 때 빠져나오는 조건이다.  
    init_task = init_task->parent;  
}
```

실습: 백대 코드(list_tasks_dfs.c)

```
#include <linux/init.h>
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/sched.h>

void dfs(struct task_struct *task) {
    struct task_struct *cursor;
    printk(KERN_INFO "COMM: %-20s STATE: %ld\tPID: %d\n", task->comm, task->state,
        task->pid);

    /* list_for_each 사용할 경우 */

    /* list_for_each_entry 사용할 경우 */
    list_for_each_entry(cursor, &task->children, sibling) {
        dfs(cursor);
    }
}

static int __init list_task_init(void) {
    // pid가 1인 프로세스 즉 init 프로세스의 pid 구조체를 가져온다.
    // 가져온 pid를 통해 해당 프로세스의 task_struct 구조체를 가져온다.

    printk(KERN_INFO "INSTALL: list_tasks_dfs\n");

    //pid가 1이 아닌 경우 프로세스의 조상을 찾아간다.

    dfs(init_task); // 깊이 우선 탐색을 통해 init 프로세스를 기점으로 모든
    // 자식노드들의 프로세스를 탐색한다.
    return 0;
}

static void __exit list_task_exit(void) {
    printk(KERN_INFO "REMOVE: list_tasks_dfs\n");
}

module_init(list_task_init);
module_exit(list_task_exit);

MODULE_LICENSE("GPL");
MODULE_DESCRIPTION("list tasks by dfs");
MODULE_AUTHOR("OS2019");
```


List_tasks_dfs(결과화면)

```
[38040.136660] INSTALL: list_tasks_dfs
[38040.136664] COMM: systemd          STATE: 1      PID: 1
[38040.136665] COMM: systemd-journal        STATE: 0      PID: 286
[38040.136666] COMM: systemd-udev          STATE: 1      PID: 302
[38040.136667] COMM: systemd-udev          STATE: 0      PID: 9513
[38040.136669] COMM: systemd-resolve       STATE: 1      PID: 409
[38040.136670] COMM: acpid                 STATE: 1      PID: 565
[38040.136671] COMM: ModemManager          STATE: 1      PID: 569
[38040.136673] COMM: snapd                 STATE: 1      PID: 574
[38040.136674] COMM: accounts-daemon       STATE: 1      PID: 575
[38040.136675] COMM: avahi-daemon          STATE: 1      PID: 578
[38040.136677] COMM: avahi-daemon          STATE: 1      PID: 596
[38040.136678] COMM: rsyslogd              STATE: 1      PID: 579
[38040.136679] COMM: cron                  STATE: 1      PID: 598
[38040.136681] COMM: dbus-daemon           STATE: 1      PID: 601
[38040.136682] COMM: wpa_supplicant        STATE: 1      PID: 672
[38040.136683] COMM: NetworkManager        STATE: 1      PID: 673
[38040.136684] COMM: dhclient              STATE: 1      PID: 6476
[38040.136686] COMM: systemd-logind        STATE: 1      PID: 678
[38040.136687] COMM: networkd-dispatcher    STATE: 1      PID: 679
[38040.136689] COMM: udisksd               STATE: 1      PID: 681
[38040.136690] COMM: polkitd               STATE: 1      PID: 692
[38040.136691] COMM: unattended-upgr        STATE: 1      PID: 773
[38040.136692] COMM: whoopsie              STATE: 1      PID: 827
[38040.136693] COMM: kerneloops            STATE: 1      PID: 842
```

Depth-First Search

