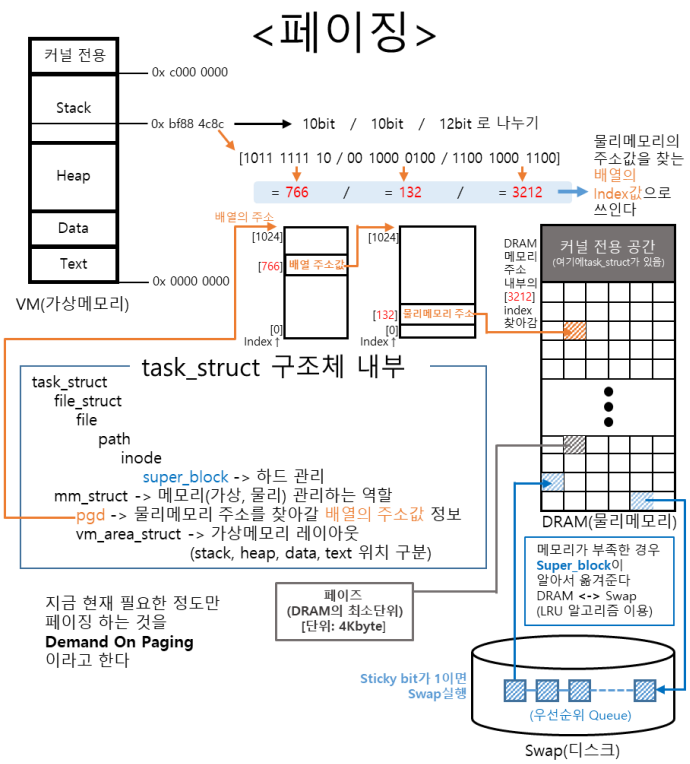
# TI DSP, MCU 및 Xilinx Zynq FPGA 프로그래밍 전문가 과정

강사 – Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 – 최대성
c3d4s19@naver.com

2018.03.24. – 022일차 수업

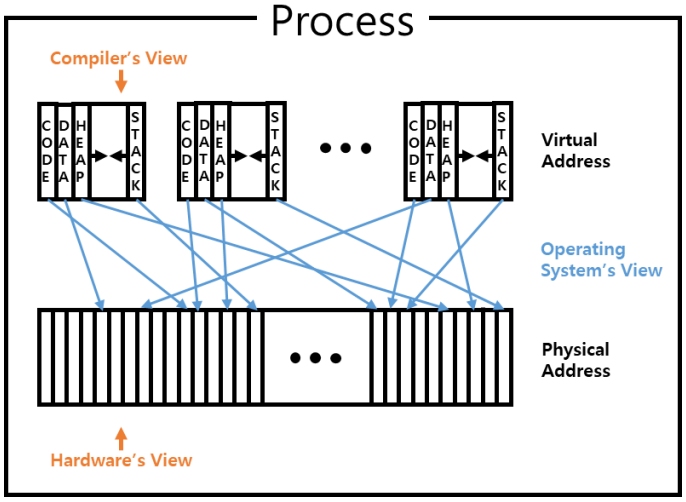
---

* 페이징




---

* Process(프로세스)




프로세스는 가상메모리 공유가 불가능하다

해결방법: IPC(Inter Process Communication)

-> 물리 메모리 공유

* 관련 함수 정리

fork()

(현재 프로세스를 모두 복사하여

별도의 자식 프로세스로 실행시킴)

-> 생성된 자식 pid(process id)값을 반환

getpid()

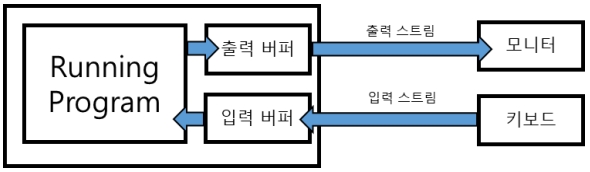
-> 자기 자신의 pid(process id)값 반환

('부모 프로세스'는 자식이 있어서 pid값이 0보다 크다

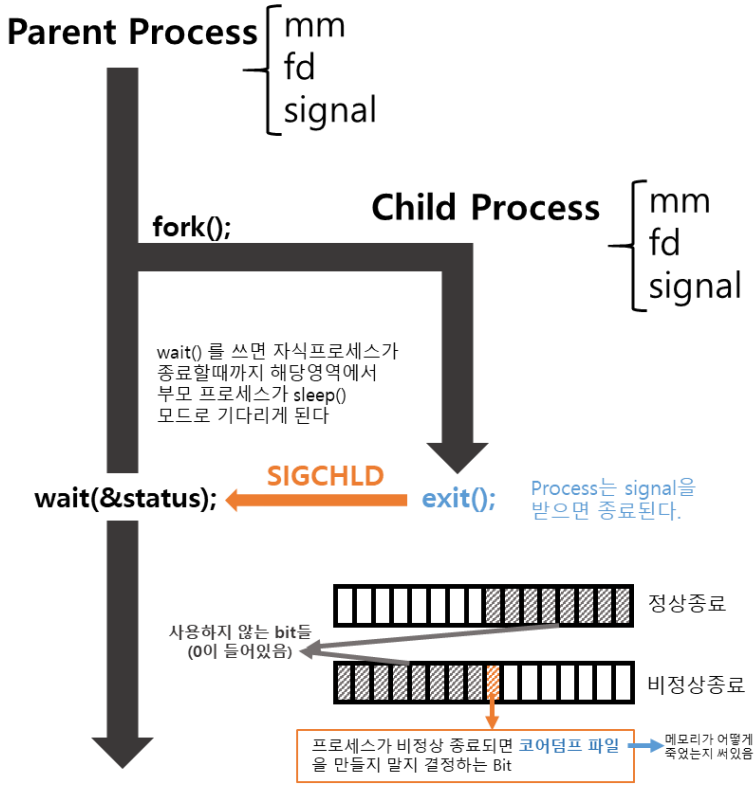
'자식 프로세스'는 자식이 없어서 pid값이 0이 된다

fflush(stdin) -> 입력버퍼 데이터 보내는 즉시 삭제

fflush(stdout) -> 출력버퍼 데이터 보내는 즉시 삭제




---



* COW(Copy On Write)

fork(); 함수 사용 후 자식 프로세스가 부모 프로세스를 즉시 모두 복사하여 실행되는 것이 아니라 필요할 때 마다 하나씩 복사해간다. (자식 프로세스가 Write 또는 부모 프로세스가 Write하는 경우)

---

* fork() 함수를 이용한 예제들

Parent process와 Child process의 pid값 비교 예제

```
#include <stdio.h>
#include <unistd.h>
#include <errno.h>
#include <stdlib.h>

int main(){
        pid_t pid;
        pid = fork();
        if(pid > 0)
                printf("parent : pid = %d, cpid
= %d₩n", getpid(), pid);
        else if(pid == 0)
                printf("child : pid = %d, cpid
= %d₩n", getpid(), pid);
        else{
                perror("fork() ");
                exit(-1);
        }
        return 0;
}
```

Parent process와 Child process의 멀티테스킹 예제1

```
#include <unistd.h>
#include <stdio.h>
#include <errno.h>
#include <stdlib.h>

int main(){
        pid_t pid;
        int i;
        pid = fork();
        if(pid > 0){
                while(1){
                        for(i = 0; i < 26; i++){
                                printf("%c ",i +
'A');
                                fflush(stdout);
                        }
                }
        }
        else if(pid == 0){
                while(1){
                        for(i = 0; i < 26; i++){
                                printf("%c ",i +
'a');
                                fflush(stdout);
                        }
                }
        }
        else{
                perror("fork() ");
                exit(-1);
        }
        printf("₩n");
        return 0;
}
```

Parent process와 Child process의 멀티테스킹 예제2

```
#include <stdio.h>
#include <unistd.h>
#include <errno.h>
#include <stdlib.h>
#include <fcntl.h>

int main(){
        int fd, ret;
        char buf[1024];
        pid_t pid;
        fd = open("myfifo", O_RDWR);
        if((pid = fork()) > 0){
                for(;;){
                        ret = read(0, buf,
sizeof(buf));
                        buf[ret] = 0;
                        printf("Keyboard : %s₩n",
buf);
                }
        }
        else if(pid == 0){
                for(;;){
                        ret = read(fd, buf,
sizeof(buf));
                        buf[ret] = 0;
                        printf("myfifo : %s₩n",
buf);
                }
        }
        else{
                perror("fork() ");
                exit(-1);
        }
        close(fd);
        return 0;
}
```

Parent process와 Child process가 복사 이후부터 task_struct를 서로 공유하지 않음을 보여주는 예제

```
#include <unistd.h>
#include <stdio.h>
#include <errno.h>
#include <stdlib.h>

int global = 100;

int main(){
        int local = 10;
        pid_t pid;
        int i;
        pid = fork();
        if(pid > 0){
                global++;
                local++;
                printf("global : %d,
local : %d₩n", global, local);
        }
        else if(pid == 0){
                printf("global : %d,
local : %d₩n", global, local);
        }
        else{
                perror("fork() ");
                exit(-1);
        }
        printf("₩n");
        return 0;
}
```

wait(&status)함수로 Child process 종료원인 출력 예제

```
#include <stdio.h>
#include <unistd.h>
#include <errno.h>
#include <stdlib.h>
#include <fcntl.h>
#include <sys/types.h>
#include <sys/wait.h>

int main(){
        pid_t pid;
        int status;
        if((pid =fork()) > 0){
                wait(&status);
                printf("status : 0x%x₩n", status
& ~(1<<7));
        }
        else if(pid == 0)
                abort();
        else{
                perror("fork() ");
                exit(-1);
        }
        return 0;
}
```