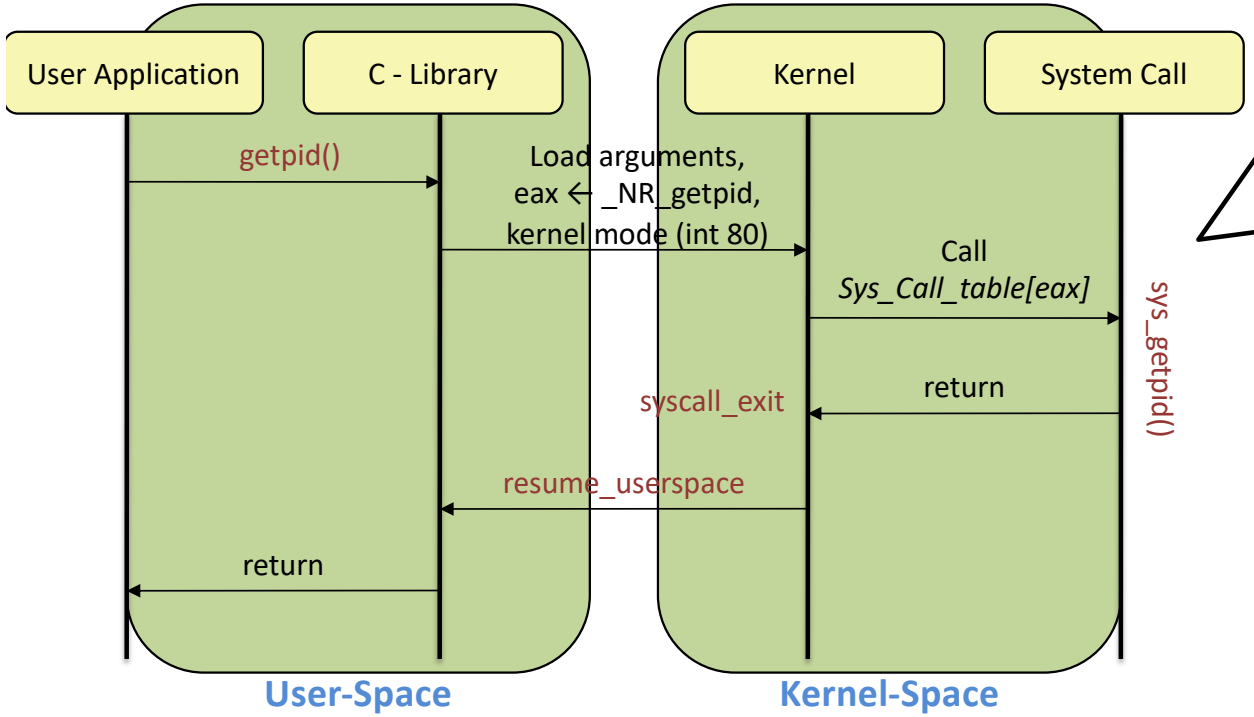
XV6 system call 관련 참고자료

System Call Interface in xv6

- System calls made by C/C++ lib functions
  - 0x80 is used for SW system calls in X86 based Intel/AMD .



syscall.h

```
// System call numbers
#define SYS_fork 1
#define SYS_exit 2
#define SYS_wait 3
#define SYS_pipe 4
#define SYS_read 5
#define SYS_kill 6
#define SYS_exec 7
#define SYS_fstat 8
#define SYS_chdir 9
#define SYS_dup 10
#define SYS_getpid 11
#define SYS_sbrk 12
#define SYS_sleep 13
#define SYS_uptime 14
#define SYS_open 15
#define SYS_write 16
#define SYS_mknod 17
#define SYS_unlink 18
#define SYS_link 19
#define SYS_mkdir 20
#define SYS_close 21
```

System Call Interface in xv6

usys.S

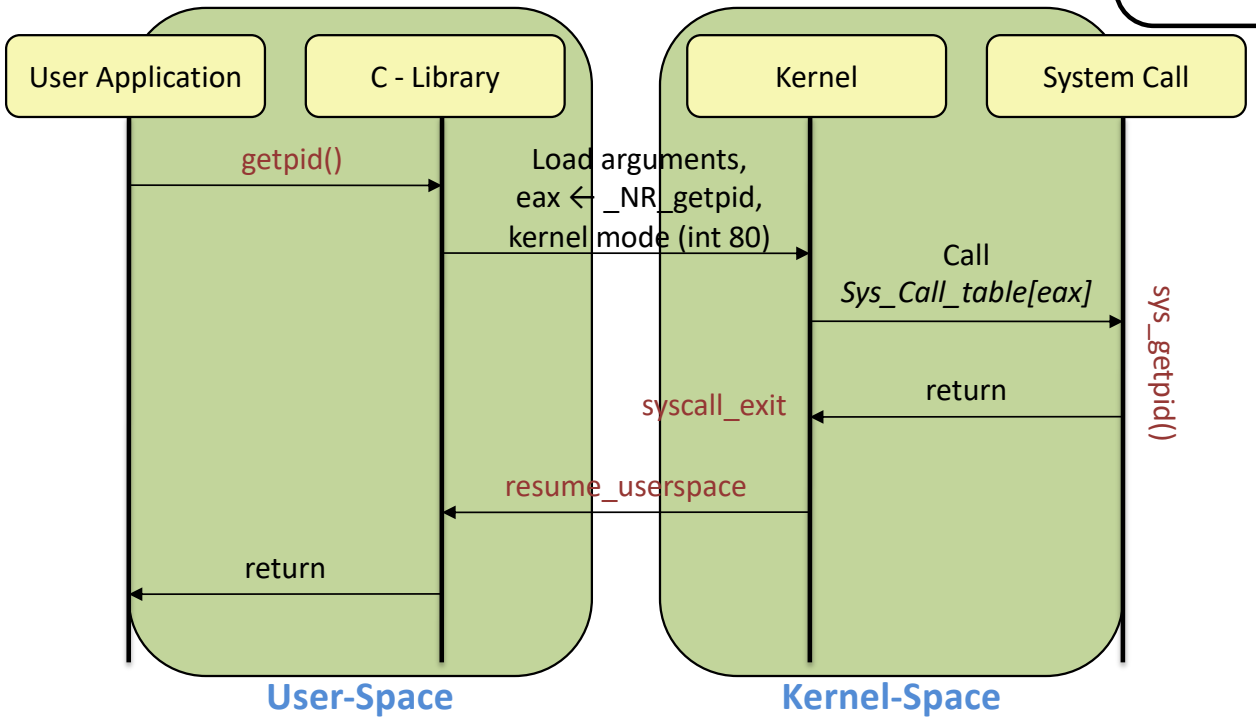
```
#include "syscall.h"
#include "traps.h"

#define SYSCALL(name)
\ .globl name; \
name: \
movl $SYS_ ## name,
%eax; \ int
$T_SYSCALL; \
ret

SYSCALL(fork)
SYSCALL(exit)
SYSCALL(wait)
SYSCALL(pipe)
SYSCALL(read)
SYSCALL(write)
SYSCALL(close)
SYSCALL(kill)
SYSCALL(exec)
SYSCALL(open)
SYSCALL(mknod)
SYSCALL(unlink)
SYSCALL(fstat)
SYSCALL(link)
SYSCALL(mkdir)
SYSCALL(chdir)
SYSCALL(dup)
SYSCALL(getpid)
SYSCALL(sbrk)
SYSCALL(sleep)
SYSCALL(uptime)
```

```
.globl fork; \
fork : \
  movl $SYS_fork, %eax; \
  int $T_SYSCALL; \
ret
```

```
.globl fork; \
fork : \
  movl $1, %eax; \
  int $64; \
ret
```



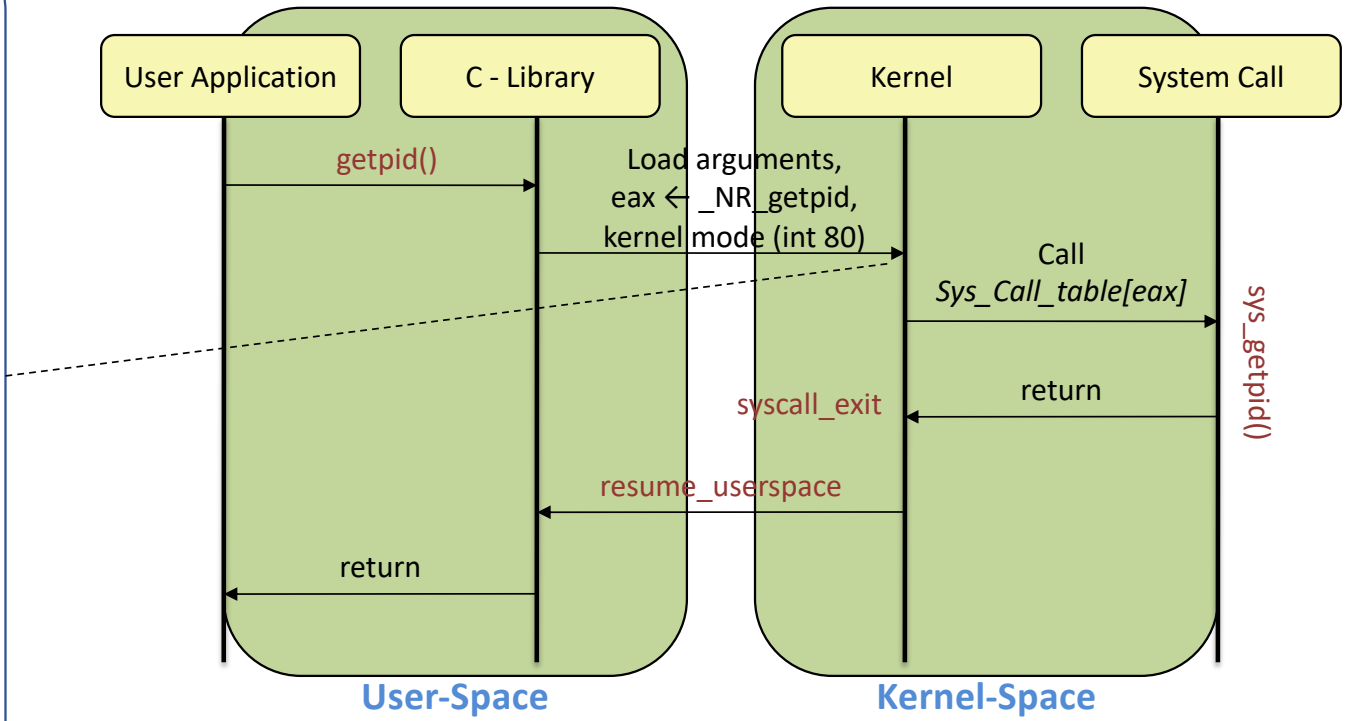
System Call Interface in xv6

trapasm.S

```
.globl alltraps
alltraps:
# Build trap frame.
pushl %ds
pushl %es
pushl %fs
pushl %gs
pushal

        .
        .
        .

# Call trap(tf), where
tf=%esp
pushl %esp
call trap
        .
        .
        .
```



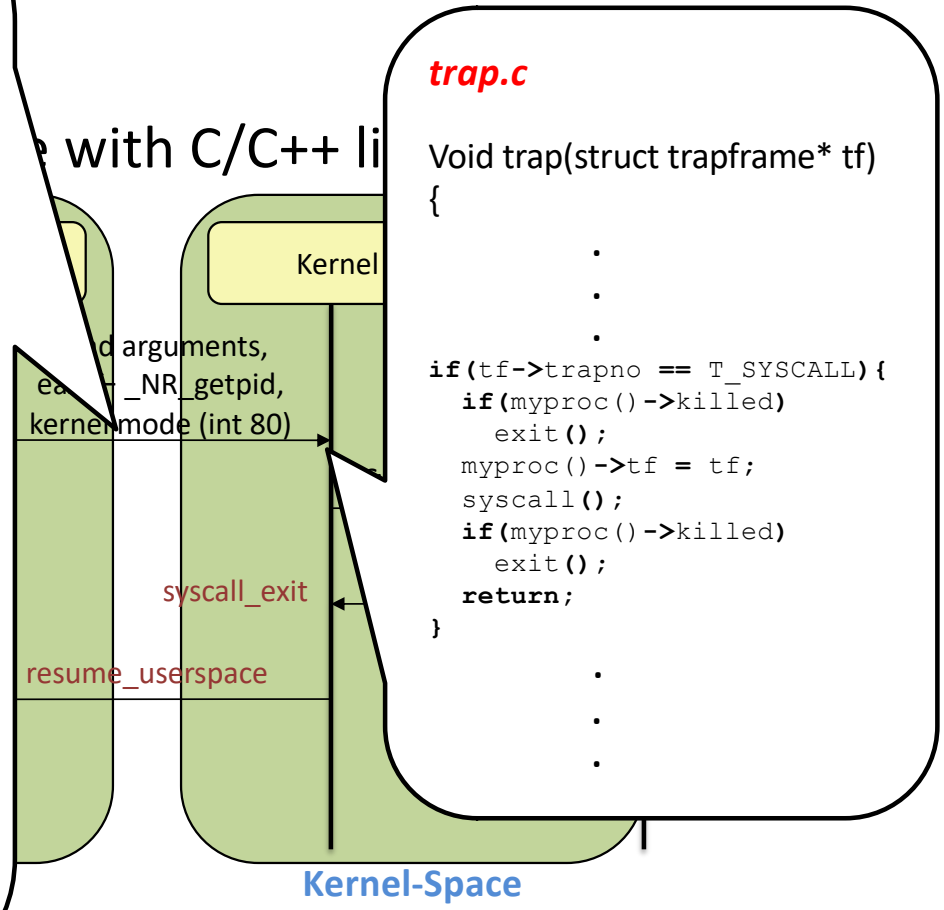
System Call Interface in xv6

syscall.c

```
static int (*syscalls[])(void) = {
[SYS_fork] sys_fork,
[SYS_exit] sys_exit,
          .
          .
          .
[SYS_close] sys_close,
};

void syscall(void) {
  int num;
  struct proc *curproc = myproc();

  num = curproc->tf->eax;
  if(num > 0 && num < NELEM(syscalls) &&
     syscalls[num]) {
     curproc->tf->eax = syscalls[num]();
  } else {
    cprintf("%d %s: unknown sys call %d\n",
     curproc->pid, curproc->name, num);
    curproc->tf->eax = -1;
  }
}
```



trap.c

```
Void trap(struct trapframe* tf)
{
          .
          .
          .
if(tf->trapno == T_SYSCALL){
  if(myproc()->killed)
    exit();
  myproc()->tf = tf;
  syscall();
  if(myproc()->killed)
    exit();
  return;
}
          .
          .
          .
```

although newer techniques for "faster" control transfer are provided by both AMD's and Intel's architecture.

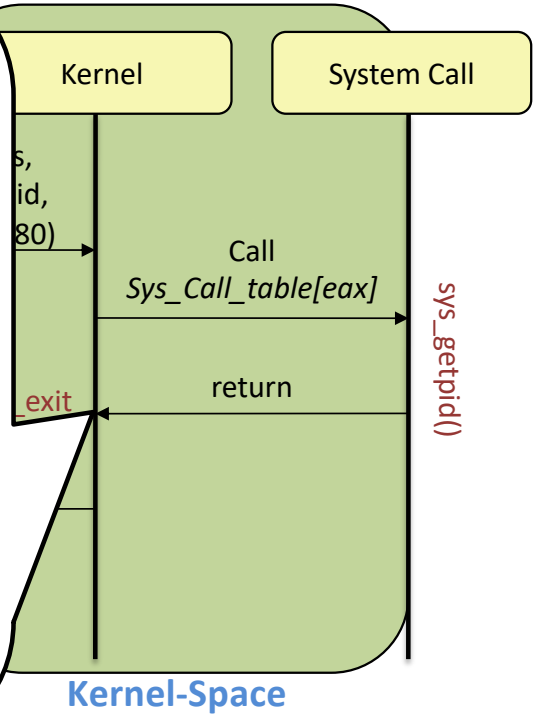
System Call Interface in xv6

Calls are usually made with C/C++ library functions

trapasm.S

```
        .
        .
        .
addl $4, %esp

# Return falls through to trapret...
.globl trapret
trapret:
  popal
  popl %gs
  popl %fs
  popl %es
  popl %ds
  addl $0x8, %esp # trapno and errcode
  iret
```



Remark: Invoking int 0x80 is common although newer techniques for "faster" control transfer are provided by both AMD's and Intel's architecture.

Process Control Block

proc.h

```
enum procstate { UNUSED, EMBRYO, SLEEPING, RUNNABLE, RUNNING, ZOMBIE };

// Per-process state
struct proc {
  uint sz;                     // Size of process memory (bytes)
  pde_t* pgdir;                // Page table
  char *kstack;                // Bottom of kernel stack for this process
  enum procstate state;        // Process state
  int pid;                     // Process ID
  struct proc *parent;         // Parent process
  struct trapframe *tf;        // Trap frame for current syscall
  struct context *context;     // swtch() here to run process
  void *chan;                  // If non-zero, sleeping on chan
  int killed;                  // If non-zero, have been killed
  struct file *ofile[NOFILE];  // Open files
  struct inode *cwd;           // Current directory
  char name[16];               // Process name (debugging)
};
```

Adding a System call in XV6 (**참고 사이트**)

- https://jehwanyoo.net/2020/10/19/xv6%EC%9D%98-%EC%8B%9C%EC%8A%A4%ED%85%9C-%EC%BD%9C-%ED%98%B8%EC%B6%9C-%EA%B3%BC%EC%A0%95/
- https://intrepidgeeks.com/tutorial/add-new-system-calls-and-user-programs-to-xv6
- https://m.blog.naver.com/PostView.naver?blogId=csi468_&logNo=221432731547&proxyReferer=