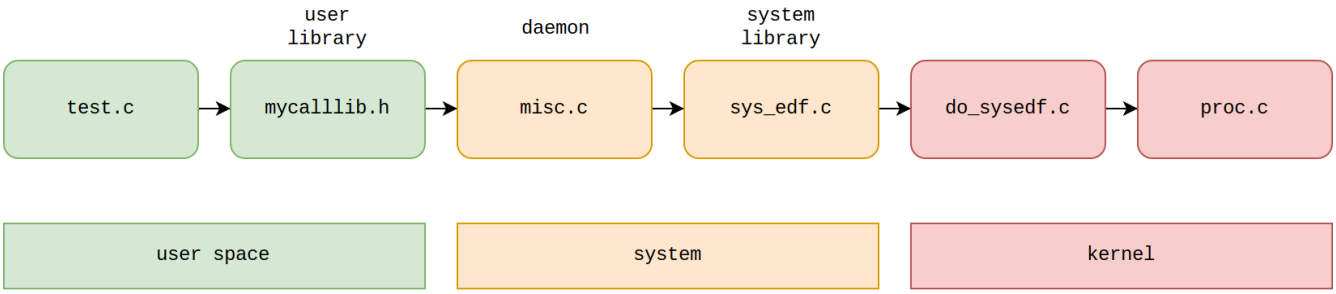
# How to Add a Kernel Call in MINIX 3.2.1




## Defining the Kernel Call number

Add the call number for sys_setedf to the call vector and increment its dimension:
/usr/src/include/minix/com.h

```
/*===========================================================================*
 *                   SYSTASK request types and field names                   *
 *===========================================================================*/

/* System library calls are dispatched via a call vector, so be careful when
 * modifying the system call numbers. The numbers here determine which call
 * is made from the call vector.
 */
#define KERNEL_CALL     0x600 /* base for kernel calls to SYSTEM */
...
#  define SYS_UPDATE     (KERNEL_CALL + 52)     /* sys_update() */
#  define SYS_EXIT       (KERNEL_CALL + 53)     /* sys_exit() */

#  define SYS_SCHEDCTL (KERNEL_CALL + 54) /* sys_schedctl() */
#  define SYS_STATECTL (KERNEL_CALL + 55) /* sys_statectl() */
#  define SYS_SAFEMEMSET (KERNEL_CALL + 56)     /* sys_safememset() */
#  define SYS_SETEDF    (KERNEL_CALL + 57) /* sys_setedf() */

/* Total */
#define NR_SYS_CALLS    58    /* number of kernel calls */
...
```

## Defining the Kernel Call prototype function

Add the prototype of your kernel function do_setedf() in the file: /usr/src/kernel/system.h

```
...
int do_schedule(struct proc * caller, message *m_ptr);
int do_schedctl(struct proc * caller, message *m_ptr);
int do_setedf(struct proc * caller, message * m_ptr);

int do_statectl(struct proc * caller, message *m_ptr);
...
```

**Mapping the Kernel Call to a prototype function**
Map SYS_SETEDF to do_setedf() in the system call table: /usr/src/kernel/system.c

```
/*===========================================================================*
 *                        initialize                                         *
 *===========================================================================*/
void system_init(void)
{
...
  /* Scheduling */
  map(SYS_SCHEDULE, do_schedule);   /* reschedule a process */
  map(SYS_SCHEDCTL, do_schedctl);   /* change process scheduler */
  map(SYS_SETEDF, do_setedf); /*set deadline to a process a process */
}
...
```

**Modify the Kernel Process Structure**
Add an integer parameter called deadline to the proc struct in /usr/src/kernel/proc.h

```
...
struct proc {
  struct stackframe_s p_reg;  /* process' registers saved in stack frame */
  struct segframe p_seg;      /* segment descriptors */
  proc_nr_t p_nr;       /* number of this process (for fast access) */
  struct priv *p_priv;        /* system privileges structure */
  volatile u32_t p_rts_flags; /* process is runnable only if zero */
  volatile u32_t p_misc_flags;      /* flags that do not suspend the process */

  int deadline; /*current process deadline*/
  char p_priority;            /* current process priority */
  u64_t p_cpu_time_left;      /* time left to use the cpu */
...
```

**Implementing the Kernel Call**
Write the implementation of do_setedf() in its own source file: /usr/src/kernel/system/do_setedf.c. The attribute m1_i3 contains the process endpoint (It will be established later on).

```
#include "kernel/system.h"
#include <minix/endpoint.h>

int do_setedf(struct proc * caller, message * m_ptr){
      struct proc *p;
      int proc_nr = 0;
      if (!isokendpt(m_ptr->m1_i3, &proc_nr))
            return EINVAL;

      p = proc_addr(proc_nr);
      p->deadline = m_ptr->m1_i2;

      printf("do_setedf.c. %d\n",m_ptr->m1_i2);

      return(OK);
}
```

**Modify System Makefile**
Add do_setedf.c to the Makefile for compilation: /usr/src/kernel/system/Makefile.inc

```
# Makefile for system library implementation
.include <bsd.own.mk>

.PATH:      ${.CURDIR}/system
SRCS+=      \
      do_fork.c \
      do_exec.c \
...
      do_setedf.c \
      do_statectl.c

.if ${MACHINE_ARCH} == "i386"
SRCS+=  \
      do_devio.c \
      do_vdevio.c
.endif
```

**Compiling the Kernel Call**
Compile the Kernel Call and include it in the OS boot image by completing the following steps:

In /usr/src/releasetools compile a new kernel image:

```
# make hdboot
```

Sync & Shutdown Minix and start it to run the new version of the OS

**Creating a system-level library interface for the Kernel Call**
Add a prototype for the sys_edf function in the file: /usr/src/include/minix/syslib.h. This will be the system library.

```
/*===========================================================================*
 * Minix system library.                                                     *
 *===========================================================================*/
int _taskcall(endpoint_t who, int syscallnr, message *msgptr);
int _kernel_call(int syscallnr, message *msgptr);
...
int sys_schedule(endpoint_t proc_ep, int priority, int quantum, int
      cpu);
int sys_schedctl(unsigned flags, endpoint_t proc_ep, int priority, int
      quantum, int cpu);
int sys_edf(int deadline,endpoint_t endpoint);
...
```

**Implementing the system-level library**
Write your implementation of the function sys_edf in a new file: /usr/src/lib/libsys/sys_edf.c.

```
#include "syslib.h"

int sys_edf(int deadline,endpoint_t endpoint){
	message m;
	m.m1_i2 = deadline;
	m.m1_i3 = endpoint;
	printf("sys_edf.c %d\n", deadline);
	return (_kernel_call(SYS_SETEDF, &m));
}
```

**Modifying the system-level library Makefile**
Add sys_edf.c to the /usr/src/lib/libsys/Makefile

```
# Makefile for libsys
.include <bsd.own.mk>

LIB=		sys

.include "arch/${MACHINE_ARCH}/Makefile.inc"

SRCS+=	\
	alloc_util.c \
...
	sys_cprof.c \
	sys_edf.c \
	sys_endsig.c \
	sys_exec.c \
...
```

**Modifying the System Tab**
Add the SYS_SETEDF service to the system tab: /usr/src/commands/service/parse.c;

```
struct
{
	char *label;
	int call_nr;
} system_tab[]=
{
	{ "PRIVCTL",		SYS_PRIVCTL },
...
	{ "EDF",		SYS_SETEDF },
	{ NULL,		0 }
};
```

**Build the updated system library and install**
In /usr/src/lib/libsys

```
# make
# make install
```

**Build the updated System Tab and install**
In /usr/src/commands/service/

```
# make
# make install
```

Sync & Shutdown Minix and start it to run the new version of the OS

**Update the System Call**
In your previously created system call invoke the sys_edf system function passing the deadline parameter and the a process endpoint (otherwise follow the *Add System Call to Minix Tutorial - Updated*)

In the file /usr/src/servers/pm/misc.c update your function

```
/*===========================================*
 *                    do_mycall                    *
 *===========================================*/
int do_mycall()
{
      int deadline = m_in.m1_i2;
      printf("misc.c %d\n", deadline);
      sys_edf(deadline, mp->mp_endpoint);
      return 0;
}
```

**Compiling the System Call**
Compile the System Call and include it in the OS boot image by completing the following steps:

In /usr/src/releasetools compile and install services:

```
# make services
# make install
```

Sync & Shutdown Minix and start it to run the new version of the OS

**Updating a user-level library interface for the System Call**
Modify your `mycalllib.h` so it prints its filename and deadline value:

```
#include <lib.h>
#include <unistd.h>

int mycall(int deadline){
      message m;
      m.m1_i2 = deadline;
      printf("mycalllib.h %d\n", deadline);
      return (_syscall(PM_PROC_NR, MYCALL, &m));
}
```

**Testing your kernel call**
Under `/home` create a file named `test_kernel_call.c`

```
#include <stdio.h>
#include <mycalllib.h>

int main(){
      int deadline=99;
      printf("test_kernel_call.c %d\n", deadline);
      mycall(deadline);
      return 0;
}
```

Compile and run with:

```
$ cc -o test_kernel_call test_kernel_call.c
$ ./test_kernel_call
```

```
# ./test_kernel_call
test_kernel_call.c 99
mycalllib.h 99
misc.c 99
sys_edf.c 99
do_setedf.c 99
# _
```