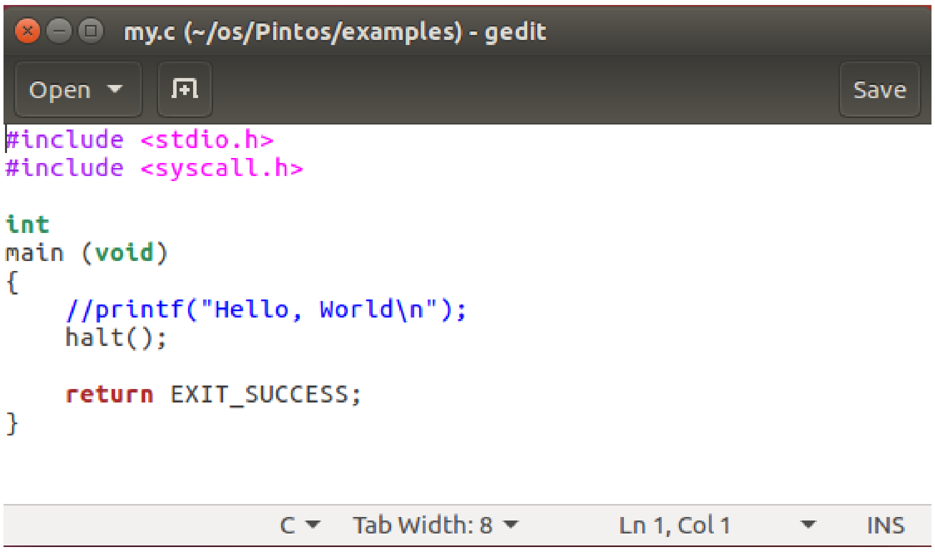
How to test system call?

Please go through file ‘Pintos/lib/user/syscall.c’, which detailed the links between system calls and the system call number.

For example, to test SYS_HALT, you can design an application my.c which call system call ‘halt()’.

1. $gedit ../../examples/my.c



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

int
main (void)
{
    //printf("Hello, World\n");
    halt();

    return EXIT_SUCCESS;
}
```

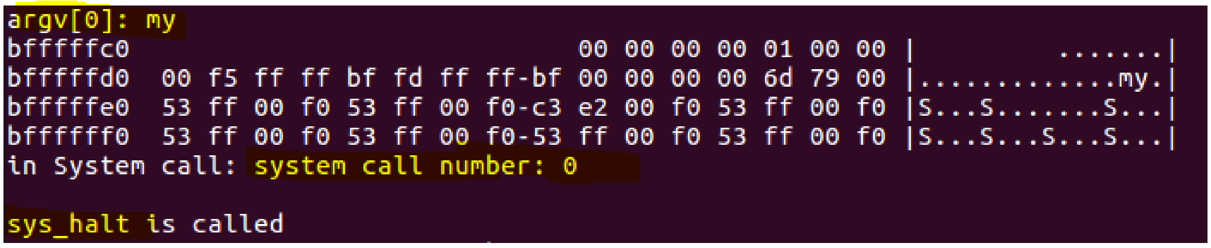
2. Build my.c in Pintos/example/

   make

3. Go to ‘Pintos/userprog/build’ and Copy the ‘my’ to pintos file system
   $pintos -p ../../examples/my -a my -- -q

4. Then run your application
   $pintos run ‘my’ -q

   you will see



```
argv[0]: my
bffffc0                            00 00 00 00 01 00 00 |          .......|
bfffffd0  00 f5 ff ff bf fd ff ff-bf 00 00 00 00 6d 79 00 |.............my.|
bfffffe0  53 ff 00 f0 53 ff 00 f0-c3 e2 00 f0 53 ff 00 f0 |S...S.......S...|
bffffff0  53 ff 00 f0 53 ff 00 f0-53 ff 00 f0 53 ff 00 f0 |S...S...S...S...|
in System call: system call number: 0

sys_halt is called
```

Similarly, you can test ‘exit, exec, wait, create, …’ in the application.

Details of file ‘Pintos/lib/user/syscall.c’ as:

```
void
halt (void)
{
  syscall0 (SYS_HALT);
  NOT_REACHED ();
}

void
exit (int status)
{
  syscall1 (SYS_EXIT, status);
  NOT_REACHED ();
}

pid_t
exec (const char *file)
{
  return (pid_t) syscall1 (SYS_EXEC, file);
}

int
wait (pid_t pid)
{
  return syscall1 (SYS_WAIT, pid);
}
```

```
bool
create (const char *file, unsigned initial_size)
{
  return syscall2 (SYS_CREATE, file, initial_size);
}

bool
remove (const char *file)
{
  return syscall1 (SYS_REMOVE, file);
}

int
open (const char *file)
{
  return syscall1 (SYS_OPEN, file);
}

int
filesize (int fd)
{
  return syscall1 (SYS_FILESIZE, fd);
}

int
read (int fd, void *buffer, unsigned size)
{
  return syscall3 (SYS_READ, fd, buffer, size);
}

int
```

```
write (int fd, const void *buffer, unsigned size)
{
  return syscall3 (SYS_WRITE, fd, buffer, size);
}

void
seek (int fd, unsigned position)
{
  syscall2 (SYS_SEEK, fd, position);
}

unsigned
tell (int fd)
{
  return syscall1 (SYS_TELL, fd);
}

void
close (int fd)
{
  syscall1 (SYS_CLOSE, fd);
}

mapid_t
mmap (int fd, void *addr)
{
  return syscall2 (SYS_MMAP, fd, addr);
}

void
munmap (mapid_t mapid)
```

```
{
  syscall1 (SYS_MUNMAP, mapid);
}

bool
chdir (const char *dir)
{
  return syscall1 (SYS_CHDIR, dir);
}

bool
mkdir (const char *dir)
{
  return syscall1 (SYS_MKDIR, dir);
}

bool
readdir (int fd, char name[READDIR_MAX_LEN + 1])
{
  return syscall2 (SYS_READDIR, fd, name);
}

bool
isdir (int fd)
{
  return syscall1 (SYS_ISDIR, fd);
}

int
inumber (int fd)
{
  return syscall1 (SYS_INUMBER, fd);
}
```