



BAHIRDAR UNIVERSITY INSTITUTE OF TECHNOLOGY

DEPARTMENT: Software engineering

COURSE: Operating system and system programming

Individual assignment:

Name: Biruktawit Chernet

ID: 1306097

System call

```
int setns(int fd, int nstype)
```

SUBMISSION DATE: 17/11/2014 EC

SUBMITTED TO: Wendmu .B

`int setns(int fd, int nstype)`

1, what/why/how/ `int setns(int fd, int nstype)`?

✦ what is `int setns(int fd, int nstype)`?

The `setns()` is a Linux-specific system call which allows the calling thread to move into different namespaces. keeping a namespace open when it contains no processes is of course only useful if we intend to later add processes to it. That is the task of the `setns()` system call, which allows the calling process to join an existing namespace.

✦ Why do we use `int setns(int fd, int nstype)`

To reassociate thread with a namespace and also the `setns()` system call allows the calling process to join an existing namespace. The namespace to join is specified via a file descriptor that refers to one of the `/proc/PID/ns` files described. Using `setns()` allows us to construct a simple but useful tool: a program that joins a specified namespace and then executes a command in that namespace.

✦ How does the `int setns(int fd, int nstype)` work?

`int setns(int fd, int nstype);`

More precisely, `setns()` disassociates the calling process from one instance of a particular namespace type and re associates the process with another instance of the same namespace type.

Our program `setns()` takes two command-line arguments. The first argument is the pathname of a `/proc/PID/ns *` symbolic link (or a file that is bind mounted to one of those symbolic links). The remaining arguments are optional command-line arguments to be given to that program. The key steps in the program are the following:

```
fd = open(argv[1], O_RDONLY); /* Get descriptor for
namespace */
```

```
setns(fd, 0); /* Join that namespace */
```

2, Briefly describe about the list of parameters and flags?

★ `fd` - the `fd` argument specifies the namespace to join; it is a file descriptor that refers to one of the symbolic links in a `/proc/PID/ns` directory. That file descriptor can be obtained either by opening one of those symbolic links directly or by

opening a file that was bind mounted to one of the links. The **fd** argument is one of the following:

- a file descriptor referring to one of the magic links in a `/proc/PID/ns` directory (or a bind mount to such a link);
- a PID file descriptor.

->If **fd** refers to a `/proc/PID/ns` link, then **setns()** re associates the calling thread with the namespace associated with that link, subject to any constraints imposed by the **nstype** argument. In this usage, each call to **setns()** changes just one of the caller's namespace memberships.

->if **fd** refer to a PID file descriptor, then **setns()** atomically moves the calling thread into one or more of the same namespaces as the thread referred to by **fd**. The **nstype** argument is a bit mask specified by ORing together one or more of the **CLONE_NEW*** namespace constants listed above. The caller is moved into each of the target thread's namespaces that is specified in **nstype**; the caller's memberships in the remaining namespaces are left unchanged.

★ **nstype** –the **nstype** argument allows the caller to check the type of namespace that **fd** refers to. If this argument is specified as zero, no check is performed. This can be useful if the caller already knows the namespace type, or does not care about the type. Specifying **nstype** instead as one of the **CLONE_NEW*** constants causes the kernel to verify that **fd** is a file descriptor for the corresponding namespace type. This can

be useful if, for example, the caller was passed the file descriptor via a UNIX domain socket and needs to verify what type of namespace it refers to.

★ The **nstype** argument is interpreted differently in each case. It specifies which type of namespace the calling thread may be associated with. This argument can have one of the following values.

0 Allow any type of namespace to be joined.

CLONE_NEWCGROUP – fd must refer to a cgroup namespace.

CLONE_NEWIPC – fd must refer to an IPC namespace.

CLONE_NEWNET – fd must refer to a network namespace.

CLONE_NEWNS – fd must refer to a mount namespace.

CLONE_NEWPID – fd must refer to a descendant PID namespace.

CLONE_NEWTIME – fd must refer to a time namespace.

CLONE_NEWUSER – fd must refer to a user namespace.

CLONE_NEWUTS – fd must refer to a UTS namespace.

->Specifying **nstype** as 0 suffices if the caller knows (or does not care) what type of namespace is referred to by fd.

->Specifying a nonzero value for **nstype** is useful if the caller does not know what type of namespace is referred to by fd and wants to ensure that the namespace is of a particular type.

(The caller might not know the type of the namespace referred to by fd if the file descriptor was opened by another process and, for example, passed to the caller via a UNIX domain socket.)

3, List the flags, their purpose with code implementation (give Example source code with output)

```
★ #define _GNU_SOURCE
#include <fcntl.h>
#include <sched.h>
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#define errExit(msg)
do { perror(msg); exit(EXIT_FAILURE);
    } while (0)
int main(int argc, char *argv[])
{
    int fd;
    if (argc < 3) {
        fprintf(stderr, "%s /proc/PID/ns/FILE cmd args...\n",
argv[0]);
        exit(EXIT_FAILURE);
    }
    fd = open(argv[1], O_RDONLY | O_CLOEXEC);
```

```
        if (fd == -1)
errExit("open");

        if (setns(fd, 0) == -1)
            errExit("setns");

        execvp(argv[2], &argv[2]);

        errExit("execvp");
    }
```

```
setns("/proc/PID/ns/mnt", CLONE_ALL);
```

-> now your process is in the same namespaces
(IPC,NET,UTS,PID,USER,MOUNT) as the /proc/PID/ns/mnt

->to change only one of the namespaces we use the one of the
statement listed below:

```
setns("/proc/PID/ns/mnt", CLONE_NEWNS);
```

-> Switch your current Mount namespace to the one pointed by
/proc/PID/ns/mnt

```
setns("/proc/PID/ns/ipc", CLONE_NEWIPC);
```

-> Switch your current IPC namespace to the one pointed by
/proc/PID/ns/ipc

```
setns("/proc/PID/ns/net", CLONE_NEWNET);
```

->Switch your current Network namespace to the one pointed
by /proc/PID/ns/net

```
setns("/proc/PID/ns/uts", CLONE_NEWUTS);
```

-> Switch your current UTS namespace to the one pointed by
/proc/PID/ns/uts

```
setns("/proc/PID/ns/pid", CLONE_NEWPID);
```

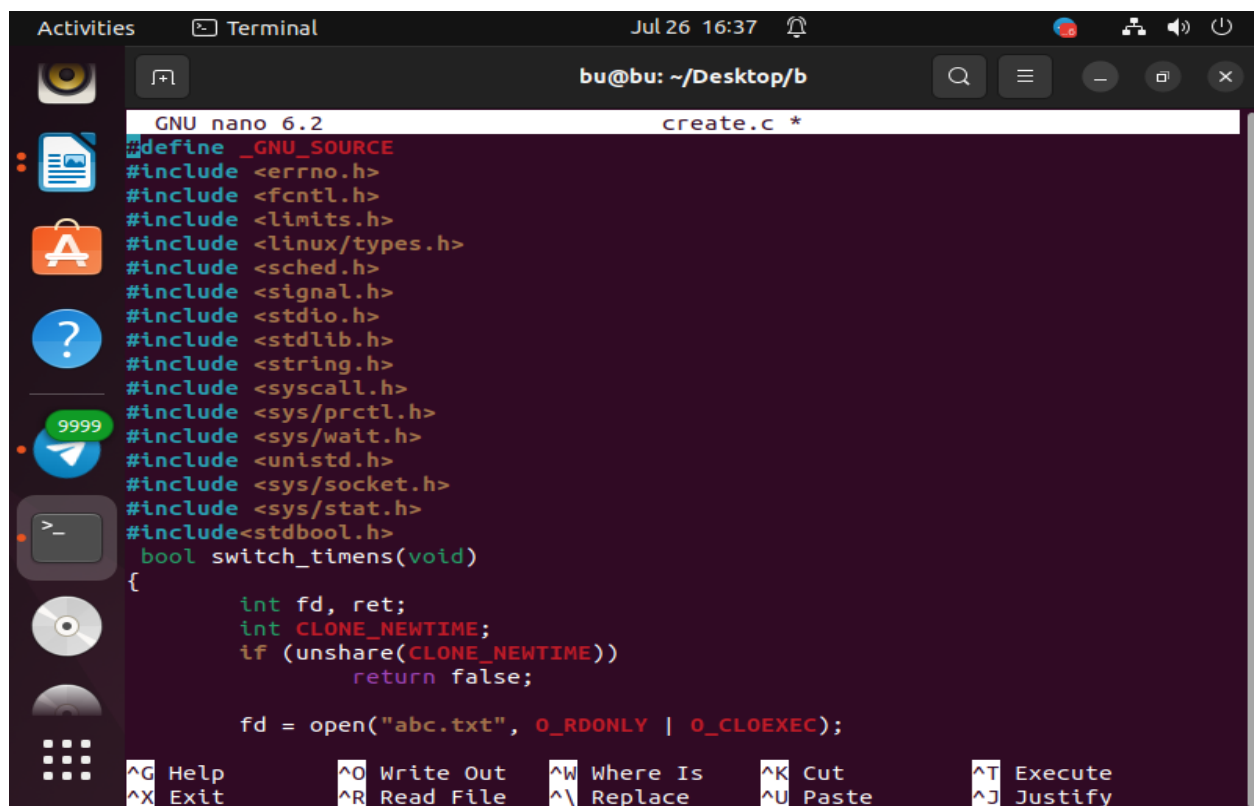
->Switch your current Pid namespace to the one pointed by
/proc/PID/ns/pid

```
setns("/proc/PID/ns/user", CLONE_NEWUSER);
```

->Switch your current User namespace to the one pointed by
/proc/PID/ns/user

```
setns("/proc/PID/ns/user", CLONE_NEWCGROUP);
```

->Switch your current Cgroup namespace to the one pointed by
/proc/PID/ns/user



```
GNU nano 6.2 create.c *
#define _GNU_SOURCE
#include <errno.h>
#include <fcntl.h>
#include <limits.h>
#include <linux/types.h>
#include <sched.h>
#include <signal.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <syscall.h>
#include <sys/prctl.h>
#include <sys/wait.h>
#include <unistd.h>
#include <sys/socket.h>
#include <sys/stat.h>
#include <stdbool.h>
bool switch_timens(void)
{
    int fd, ret;
    int CLONE_NEWTIME;
    if (unshare(CLONE_NEWTIME))
        return false;

    fd = open("abc.txt", O_RDONLY | O_CLOEXEC);
    ^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute
    ^X Exit      ^R Read File  ^\ Replace    ^U Paste      ^J Justify
```


Player ▾ || □ ✕

Activities Terminal Jul 26 16:39

bu@bu: ~/Desktop/b

GNU nano 6.2 create.c *

```
#include <unistd.h>
#include <sys/socket.h>
#include <sys/stat.h>
#include <stdbool.h>
bool switch_timens(void)
{
    int fd, ret;
    int CLONE_NEWTIME;
    if (unshare(CLONE_NEWTIME))
        return false;

    fd = open("abc.txt", O_RDONLY | O_CLOEXEC);
    if (fd < 0)
        return false;

    ret = setns(fd, CLONE_NEWTIME);
    return ret == 0;
}
close(fd);
printf("%d\n",ret);
}
int main(){
    return 0;
}
```

⌘G Help ⌘O Write Out ⌘W Where Is ⌘K Cut ⌘T Execute
⌘X Exit ⌘R Read File ⌘\ Replace ⌘U Paste ⌘J Justify

Ubuntu 64-bit - VMware Workstation 16 Player (Non-commercial use only)

Player ▾ || □ ✕

Activities Terminal Jul 26 16:52

bu@bu: ~/Desktop/b

```
bu@bu:~$ cd Desktop
bu@bu:~/Desktop$ mkdir b
bu@bu:~/Desktop$ cd b
bu@bu:~/Desktop/b$ nano create.c
bu@bu:~/Desktop/b$ gcc create.c -o rn
create.c:18:10: fatal error: pidfd.h: No such file or directory
 18 | #include "pidfd.h"
    | ~~~~~
compilation terminated.
bu@bu:~/Desktop/b$ nano create.c
bu@bu:~/Desktop/b$ gcc create.c -o rn
create.c:17:10: fatal error: ../clone3/clone3_selftests.h: No such file or directory
 17 | #include "../clone3/clone3_selftests.h"
    | ~~~~~
compilation terminated.
bu@bu:~/Desktop/b$ nano create.c
bu@bu:~/Desktop/b$ gcc create.c -o rn
create.c:18:8: error: unknown type name 'bool'
 18 | static bool switch_timens(void)
    | ~~~~~
create.c: In function 'switch_timens':
create.c:22:21: error: 'CLONE_NEWTIME' undeclared (first use in this function);
did you mean 'CLONE_NEWIPC'?
 22 |         if (unshare(CLONE_NEWTIME))
    |                     ~~~~~
create.c:22:21: note: each undeclared identifier is reported only once for each
function it appears in
```

Ubuntu 64-bit - VMware Workstation 16 Player (Non-commercial use only)

Player ▾ || ▢ ✕

Activities Terminal Jul 26 16:57 🔔 🔍 ⚙️ 🔌

bu@bu: ~/Desktop/b 🔍 ⚙️ 🔌 ✕

```
bu@bu:~/Desktop/b$ ./rn
bu@bu:~/Desktop/b$ ls
abc.txt  a.out  create.c  rn
bu@bu:~/Desktop/b$ -la
Command '-la' not found, did you mean:
  command 'tla' from deb tla (1.3.5+dfsg1-2build1)
Try: sudo apt install <deb name>
bu@bu:~/Desktop/b$
bu@bu:~/Desktop/b$ nano create.c
bu@bu:~/Desktop/b$ gcc create.c -o rn
bu@bu:~/Desktop/b$ ./rn
bu@bu:~/Desktop/b$ ls
abc.txt  a.out  create.c  rn
bu@bu:~/Desktop/b$ -la
Command '-la' not found, did you mean:
  command 'tla' from deb tla (1.3.5+dfsg1-2build1)
Try: sudo apt install <deb name>
bu@bu:~/Desktop/b$ nano create.c
bu@bu:~/Desktop/b$ gcc create.c -o rn
bu@bu:~/Desktop/b$ nano create.c
bu@bu:~/Desktop/b$ sudo touch abc.txt
[sudo] password for bu:
bu@bu:~/Desktop/b$ ls
abc.txt  a.out  create.c  rn
bu@bu:~/Desktop/b$ chmod ug=rwx abc.txt
bu@bu:~/Desktop/b$ ls
abc.txt  a.out  create.c  rn
bu@bu:~/Desktop/b$ nano create.c
bu@bu:~/Desktop/b$ gcc create.c -o rn
```