**Федеральное государственное автономное образовательное учреждение**

**Высшего профессионального образования**

Санкт-Петербургский политехнический университет

Институт информационных технологий и управления

Кафедра компьютерных систем и программных технологий

**Отчёт по лабораторной работе №8**

«Системные вызов sys\_socketcall, sys\_hostname, sys\_kill»

Работу выполнил студент гр. № 53501/3 Цыганов А.А.

Работу принял преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Душутина Е.В.

Санкт-Петербург

2016

# Цель работы:

В данной лабораторной работе необходимо:

* Исследовать механизм работы системных вызовов, используемый в Linux
* Провести описание анализируемых системных вызовов sys\_socketcall, sys\_hostname, sys\_kill: функциональность, сигнатура, возвращаемое значение.
* Проанализировать и привести примеры использования системных вызовов.

# Ход работы:

Для работы была использована виртуальная машина VMware Workstation 10.0.4, с образом ОС Linux Ubuntu 64-bit, kernel 4.2.0-16-generic.

alex@ubuntu:~/Dropbox/политех/Системное программирование/Л. р. № 7$ uname -a

Linux ubuntu 4.2.0-16-generic #19-Ubuntu SMP Thu Oct 8 15:35:06 UTC 2015 x86\_64 x86\_64 x86\_64 GNU/Linux

Исследование системного вызова sys\_socketcall, sys\_hostname, sys\_kill.

Системные вызовы – это интерфейс для прикладных программ и демонов, который позволяет им обращаться к возможностям ядра. Для того, чтобы из прикладных программ совершать системные вызовы, в стандартной библиотеке языка Си создается набор функций поверх системных вызовов. Например, для sys\_socketcall(), используется функция socket().

**Системные вызовы** (system calls) являются интерфейсом между приложениями пространства пользователя и сервисами, которые предоставляет ядро. Вызов не может быть осуществлен напрямую, так как сервисы предоставляются ядром (реализованы в ядре); вместо этого необходимо использовать процесс пересечения границы user-space/kernel.

Все объявления системных вызов находится в файле “Linux/include/linux/syscalls.h”. Ниже приведен фрагмент из этого файла, с системными функциями:

618 asmlinkage long sys\_socket(int, int, int);

374 asmlinkage long sys\_kill(int pid, int sig);

637 asmlinkage long sys\_gethostname(char \_\_user \*name, int len);

638 asmlinkage long sys\_sethostname(char \_\_user \*name, int len);

**socketcall** - системные вызовы сокетов

СИНТАКСИС

**int socketcall(int call, unsigned long \*args);**

ОПИСАНИЕ

socketcall- это общая точка входа в ядро всех системных вызовов сокетов. call задает необходимую функцию; а args указывает на структуру с параметрами функции.

Пользовательские программы должны оперировать обычными функциями, имеющими стандартные имена. О существовании socketcall необходимо знать лишь создателям стандартных библиотек и отладчикам ядра.

СООТВЕТСТВИЕ СТАНДАРТАМ

Эта функция специфична для Linux. Не рекомендуется использовать ее в программах, переносимых в другие системы.

Далее приведем пример программы, в которой используется функция для системного вызова sys\_socketcall().

#include <sys/socket.h>

int main(int argc, char\*\* argv)

{

int res = socket( 1, 2, 3 ); /\* Call socket function \*/

}

Исследуем обращение к системному вызову sys\_socketcall из функции socket(). Для этого скомпилируем файл с флагом: -static –g. И воспользуемся отладчиком gdb

gcc -static -g socktes.c

/\*

\* System call vectors.

\*

\* Argument checking cleaned up. Saved 20% in size.

\* This function doesn't need to set the kernel lock because

\* it is set by the callees.

\*/

asmlinkage long sys\_socketcall(int call, unsigned long \*args)

{

unsigned long a[6];

unsigned long a0,a1;

int err;

if(call<1||call>SYS\_RECVMSG)

return -EINVAL;

/\* copy\_from\_user should be SMP safe. \*/

if (copy\_from\_user(a, args, nargs[call]))

return -EFAULT;

a0=a[0];

a1=a[1];

switch(call)

{

case SYS\_SOCKET:

err = sys\_socket(a0,a1,a[2]);

break;

case SYS\_BIND:

err = sys\_bind(a0,(struct sockaddr \*)a1, a[2]);

break;

case SYS\_CONNECT:

err = sys\_connect(a0, (struct sockaddr \*)a1, a[2]);

break;

case SYS\_LISTEN:

err = sys\_listen(a0,a1);

break;

case SYS\_ACCEPT:

err = sys\_accept(a0,(struct sockaddr \*)a1, (int \*)a[2]);

break;

case SYS\_GETSOCKNAME:

err = sys\_getsockname(a0,(struct sockaddr \*)a1, (int \*)a[2]);

break;

case SYS\_GETPEERNAME:

err = sys\_getpeername(a0, (struct sockaddr \*)a1, (int \*)a[2]);

break;

case SYS\_SOCKETPAIR:

err = sys\_socketpair(a0,a1, a[2], (int \*)a[3]);

break;

case SYS\_SEND:

err = sys\_send(a0, (void \*)a1, a[2], a[3]);

break;

case SYS\_SENDTO:

err = sys\_sendto(a0,(void \*)a1, a[2], a[3],

(struct sockaddr \*)a[4], a[5]);

break;

case SYS\_RECV:

err = sys\_recv(a0, (void \*)a1, a[2], a[3]);

break;

case SYS\_RECVFROM:

err = sys\_recvfrom(a0, (void \*)a1, a[2], a[3],

(struct sockaddr \*)a[4], (int \*)a[5]);

break;

case SYS\_SHUTDOWN:

err = sys\_shutdown(a0,a1);

break;

case SYS\_SETSOCKOPT:

err = sys\_setsockopt(a0, a1, a[2], (char \*)a[3], a[4]);

break;

case SYS\_GETSOCKOPT:

err = sys\_getsockopt(a0, a1, a[2], (char \*)a[3], (int \*)a[4]);

break;

case SYS\_SENDMSG:

err = sys\_sendmsg(a0, (struct msghdr \*) a1, a[2]);

break;

case SYS\_RECVMSG:

err = sys\_recvmsg(a0, (struct msghdr \*) a1, a[2]);

break;

default:

err = -EINVAL;

break;

}

return err;

}

**hostname** - выдает или устанавливает имя компьютера

СИНТАКСИС

hostname [ИМЯ]

hostname ПАРАМЕТР

ОПИСАНИЕ

Выдает или устанавливает системное имя компьютера.

--help

выдает эту информацию и заканчивает работу

--version

выдает информацию о версии и заканчивает работу

**kill** - посылает сигнал процессу или выводит список допустимых сигналов

ОБЗОР

kill [-s СИГНАЛ | -СИГНАЛ] PID...

kill -l [СИГНАЛ]...

kill -t [СИГНАЛ]...

ОПИСАНИЕ

Посылает сигнал процессу или выводит список допустимых сигналов

Аргументы, обязательные для полных вариантов опций, являются обязательными также и для кратких вариантов.

-s, --signal=СИГНАЛ, -СИГНАЛ

имя или номер посылаемого сигнала

-l, --list

вывести имена сигналов или вывести имя сигнала, соответствующее номеру, и наоборот

-t, --table

вывести информацию о сигналах в виде таблицы

--help

вывести справку и закончить работу

--version

вывести информацию о версии и закончить работу

СИГНАЛ может указываться в виде имени (например, `HUP') или номера (например, `1'). Также в качестве СИГНАЛА можно указывать код выхода который программа должна сообщить системе при завершении.

PID - числовой идентификатор процесса. Если число отрицательное, оно определяет группу процесса.

Используемые источники

1. <http://welinux.ru/post/3905/>
2. <https://www.opennet.ru/man.shtml?topic=socketcall&russian=0&category=&submit=%F0%CF%CB%C1%DA%C1%D4%D8+man>
3. <https://www.opennet.ru/man.shtml?topic=kill&russian=0&category=&submit=%F0%CF%CB%C1%DA%C1%D4%D8+man>
4. <https://habrahabr.ru/post/268145/>
5. <http://lxr.free-electrons.com/source/include/linux/syscalls.h#L620>
6. http://www.skyfree.org/linux/kernel\_network/socket.html