

PB167 #2

Dušan Klinec

PB167

February 24, 2015

First step

- 1 Make sure you have: `C:\PB167\Fedora12proPB167.vdi`
If not, download it from IS study materials.
- 2 do not use this original. Copy this image to your private directory
e.g., `C:\PB167\UC0\Fedora12proPB167.vdi`
- 3 Download link: <https://is.muni.cz/auth/el/1433/jaro2015/PB167/um/zriha/cv2/Fedora12proPB167.vdi>

First step

- 1 Make sure you have: `C:\PB167\Fedora12proPB167.vdi`
If not, download it from IS study materials.
- 2 do not use this original. Copy this image to your private directory
e.g., `C:\PB167\UC0\Fedora12proPB167.vdi`
- 3 Download link: <https://is.muni.cz/auth/el/1433/jaro2015/PB167/um/zriha/cv2/Fedora12proPB167.vdi>

Windows syscall recap

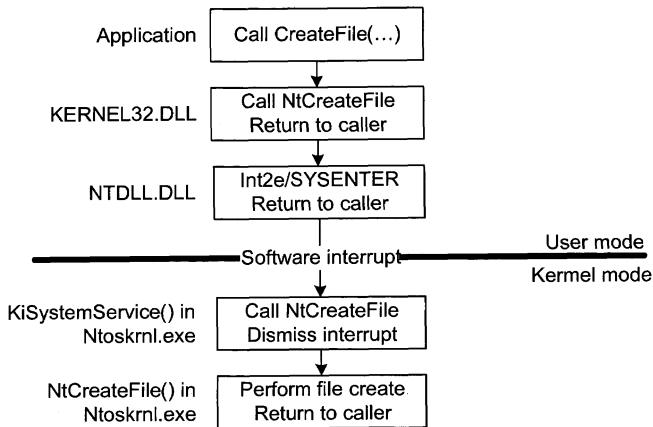
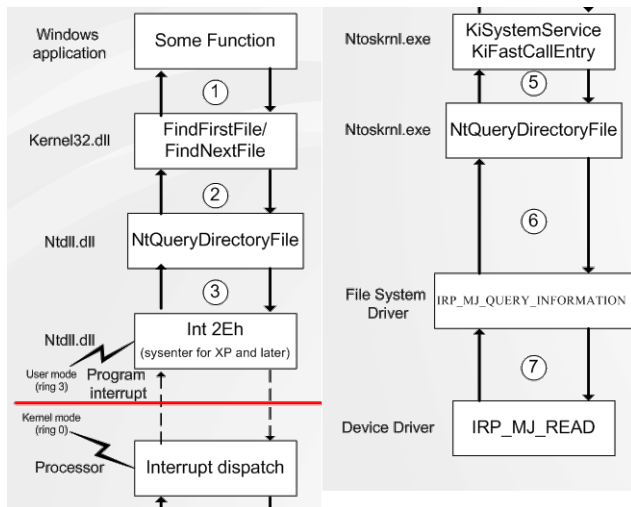


FIG. 1a

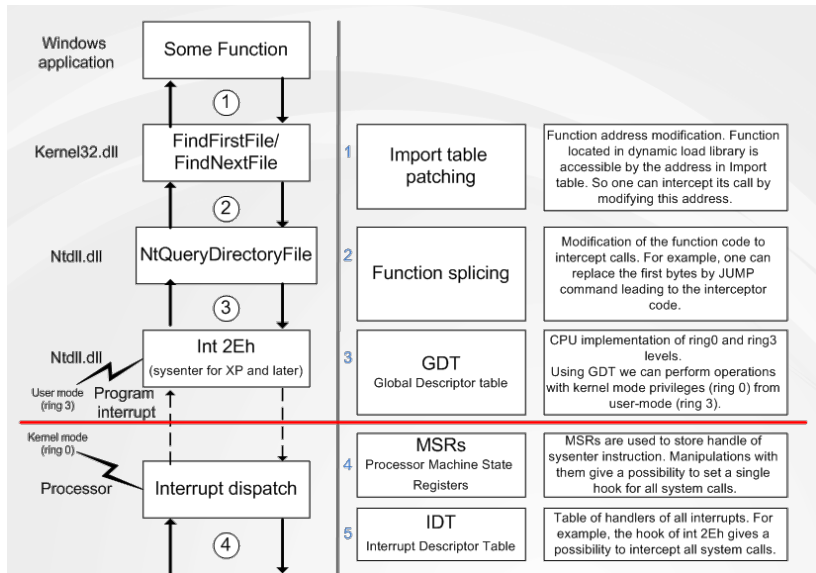
<http://www.google.co.in/patents/US7685638>

Windows syscall recap

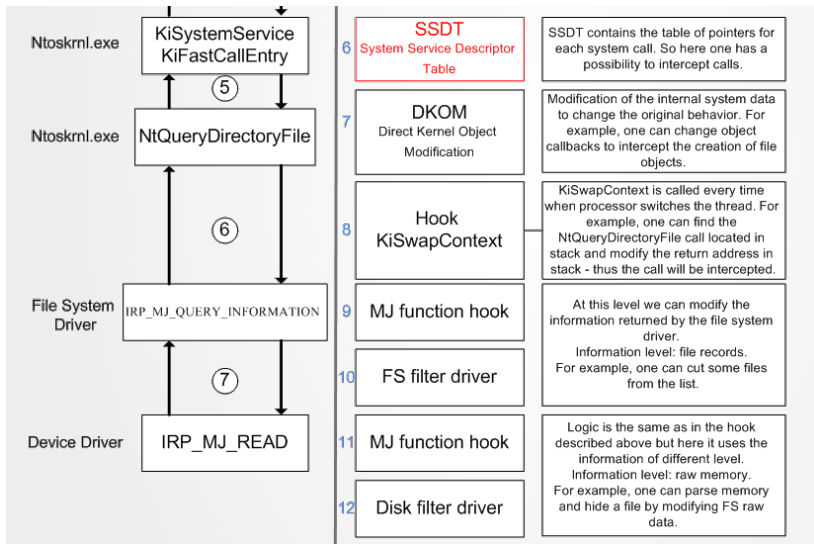


<http://www.codeproject.com/KB/winsdk/detect-driver>

Windows syscall recap

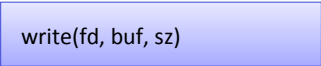


Windows syscall recap



<http://www.codeproject.com/KB/winsdk/detect-driver>

Linux syscall recap



`write(fd, buf, sz)`

User program

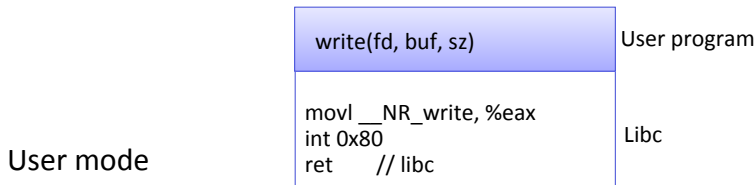
User mode

kernel mode

To find code for a Linux syscall: [hDp://syscalls.kernelgrok.com](http://syscalls.kernelgrok.com)

<http://www.cs.columbia.edu/~krj/os/lectures/L04-IPC.pdf>

Linux syscall recap

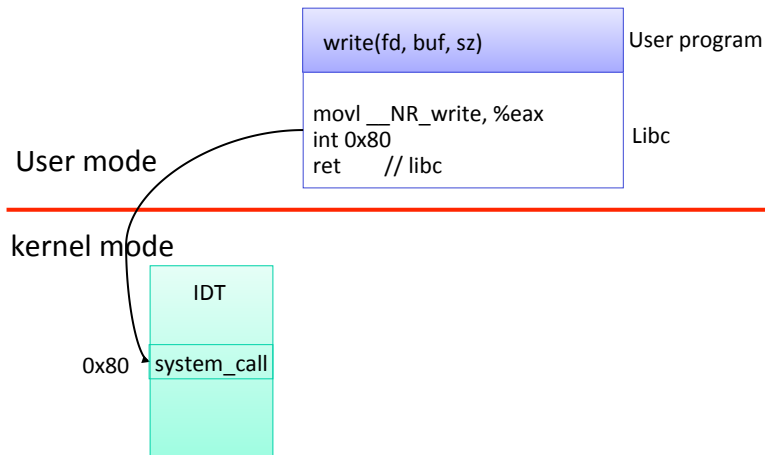


kernel mode

To find code for a Linux syscall: [hDp://syscalls.kernelgrok.com](http://syscalls.kernelgrok.com)

<http://www.cs.columbia.edu/~krj/os/lectures/L04-IPC.pdf>

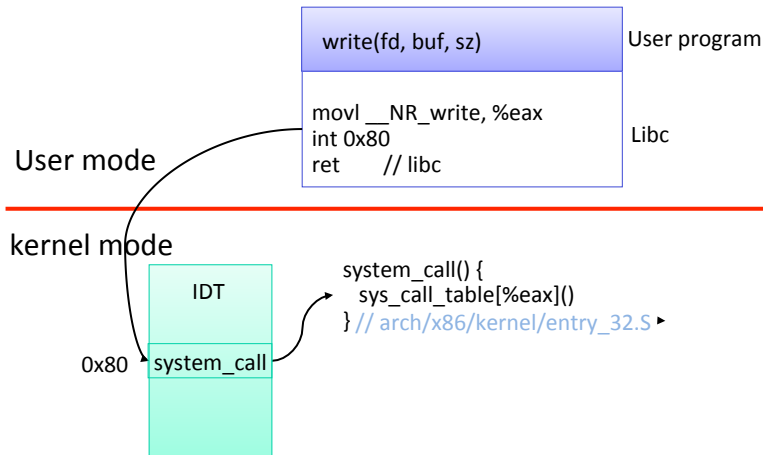
Linux syscall recap



To find code for a Linux syscall: [hDp://syscalls.kernelgrok.com](http://syscalls.kernelgrok.com)

<http://www.cs.columbia.edu/~krj/os/lectures/L04-IPC.pdf>

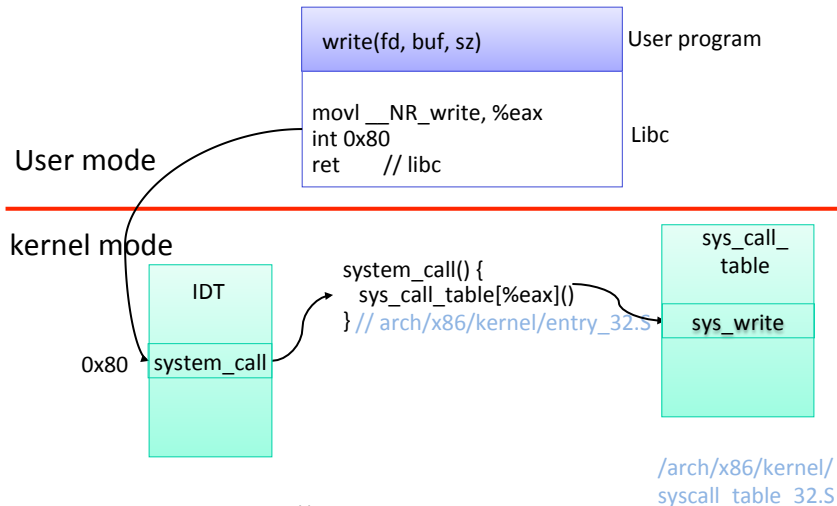
Linux syscall recap



To find code for a Linux syscall: hDp://syscalls.kernelgrok.com

<http://www.cs.columbia.edu/~krj/os/lectures/L04-IPC.pdf>

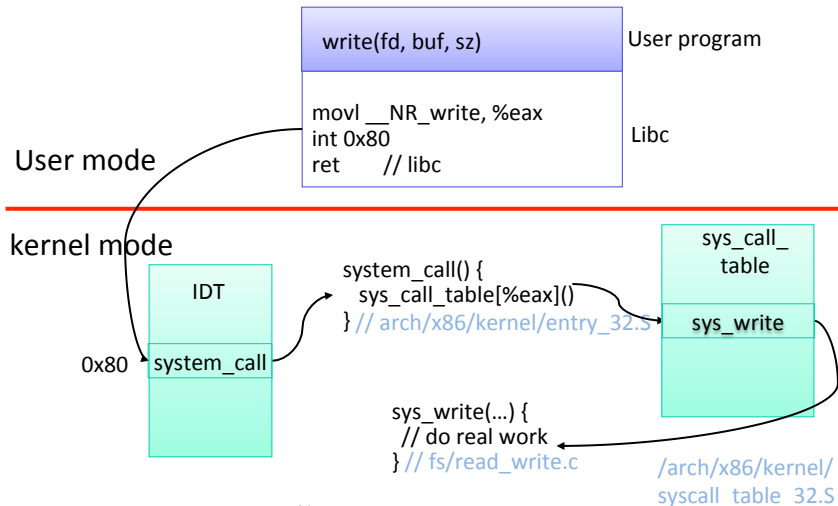
Linux syscall recap



To find code for a Linux syscall: hDp://syscalls.kernelgrok.com

<http://www.cs.columbia.edu/~krj/os/lectures/L04-IPC.pdf>

Linux syscall recap



To find code for a Linux syscall: hDp://syscalls.kernelgrok.com

<http://www.cs.columbia.edu/~krj/os/lectures/L04-IPC.pdf>

VirtualBox image

- 1 Make sure you have: `C:\PB167\Fedora12proPB167.vdi`
If not, download it from IS study materials.
- 2 do not use this original. Copy this image to your private directory
e.g., `C:\PB167\UC0\Fedora12proPB167.vdi`
- 3 Start VirtualBox, create a new virtual machine with existing drive image.
- 4 Enable PAE support. Settings → System → Processor.
- 5 Add one NATed network interface (if not already).
- 6 Boot it. user: root
password: OperacniSystemy

VirtualBox image

- 1 Make sure you have: `C:\PB167\Fedora12proPB167.vdi`
If not, download it from IS study materials.
- 2 do not use this original. Copy this image to your private directory
e.g., `C:\PB167\UC0\Fedora12proPB167.vdi`
- 3 Start VirtualBox, create a new virtual machine with existing drive image.
- 4 Enable PAE support. Settings → System → Processor.
- 5 Add one NATed network interface (if not already).
- 6 Boot it. user: root
password: OperacniSystemy

VirtualBox image

- 1 Make sure you have: `C:\PB167\Fedora12proPB167.vdi`
If not, download it from IS study materials.
- 2 do not use this original. Copy this image to your private directory
e.g., `C:\PB167\UC0\Fedora12proPB167.vdi`
- 3 Start VirtualBox, create a new virtual machine with existing drive image.
- 4 Enable PAE support. Settings → System → Processor.
- 5 Add one NATed network interface (if not already).
- 6 Boot it. user: root
password: OperacniSystemy

VirtualBox image

- 1 Make sure you have: `C:\PB167\Fedora12proPB167.vdi`
If not, download it from IS study materials.
- 2 do not use this original. Copy this image to your private directory
e.g., `C:\PB167\UC0\Fedora12proPB167.vdi`
- 3 Start VirtualBox, create a new virtual machine with existing drive image.
- 4 Enable PAE support. Settings → System → Processor.
- 5 Add one NATed network interface (if not already).
- 6 Boot it. user: root
password: OperacniSystemy

VirtualBox image

- 1 Make sure you have: `C:\PB167\Fedora12proPB167.vdi`
If not, download it from IS study materials.
- 2 do not use this original. Copy this image to your private directory
e.g., `C:\PB167\UC0\Fedora12proPB167.vdi`
- 3 Start VirtualBox, create a new virtual machine with existing drive image.
- 4 Enable PAE support. Settings → System → Processor.
- 5 Add one NATed network interface (if not already).
- 6 Boot it. user: root
password: OperacniSystemy

VM network

- 1 Look at the network interfaces: `ifconfig -a`
You should see interface with name `eth[0-9]`.
- 2 Run DHCP in your interface:
`dhclient eth2`
- 3 Test your connectivity:
`ping muni.cz`
- 4 Install additional help tools
`yum makecache`
`yum install -y openssh-clients strace ltrace rsync vim`

VM network

- 1 Look at the network interfaces: `ifconfig -a`
You should see interface with name `eth[0-9]`.
- 2 Run DHCP in your interface:
`dhclient eth2`
- 3 Test your connectivity:
`ping muni.cz`
- 4 Install additional help tools
`yum makecache`
`yum install -y openssh-clients strace ltrace rsync vim`

VM network

- 1 Look at the network interfaces: `ifconfig -a`
You should see interface with name `eth[0-9]`.
- 2 Run DHCP in your interface:
`dhclient eth2`
- 3 Test your connectivity:
`ping muni.cz`
- 4 Install additional help tools
`yum makecache`
`yum install -y openssh-clients strace ltrace rsync vim`

VM network

- 1 Look at the network interfaces: `ifconfig -a`
You should see interface with name `eth[0-9]`.
- 2 Run DHCP in your interface:
`dhclient eth2`
- 3 Test your connectivity:
`ping muni.cz`
- 4 Install additional help tools
`yum makecache`
`yum install -y openssh-clients strace ltrace rsync vim`

New syscall

- 1 Open file:

```
/root/rpmbuild/BUILD/kernel-2.6.31/linux-2.6.31.i686  
/arch/x86/include/asm/unistd_32.h
```

- 2 Add a new constant for system calls:

```
__NR_pb167_01  
__NR_pb167_02
```

Already added, do not save! Will recompile whole kernel!

- 3 Open file:

```
/root/rpmbuild/BUILD/kernel-2.6.31/linux-2.6.31.i686  
/arch/x86/kernel/syscall_table_32.S
```

- 4 Add a new system call to syscall table.

New syscall

- 1 Open file:
`/root/rpmbuild/BUILD/kernel-2.6.31/linux-2.6.31.i686`
`/arch/x86/include/asm/unistd_32.h`
- 2 Add a new constant for system calls:
`__NR_pb167_01`
`__NR_pb167_02`
Already added, do not save! Will recompile whole kernel!
- 3 Open file:
`/root/rpmbuild/BUILD/kernel-2.6.31/linux-2.6.31.i686`
`/arch/x86/kernel/syscall_table_32.S`
- 4 Add a new system call to syscall table.

New syscall

- 1 Open file:
`/root/rpmbuild/BUILD/kernel-2.6.31/linux-2.6.31.i686
/arch/x86/include/asm/unistd_32.h`
- 2 Add a new constant for system calls:
`__NR_pb167_01
__NR_pb167_02`
Already added, do not save! Will recompile whole kernel!
- 3 Open file:
`/root/rpmbuild/BUILD/kernel-2.6.31/linux-2.6.31.i686
/arch/x86/kernel/syscall_table_32.S`
- 4 Add a new system call to syscall table.

New syscall

Offset	Symbol	sys_call_table	System call location
0	__NR_restart_syscall	long sys_restart_syscall	--> ./linux/kernel/signal.c
4	__NR_exit	long sys_exit	--> ./linux/kernel/exit.c
8	__NR_exit	long sys_fork	--> ./linux/arch/386/kernel/process.c
1272	__NR_getcpu	long sys_getcpu	--> ./linux/kernel/sys.c
1276	__NR_epoll_pwait	long sys_epoll_pwait	--> ./linux/kernel/sys_ni.c
	__NR_syscalls	-----	

Dashed arrows indicate the source of the symbols:

 - An arrow points from `./linux/include/asm/unistd.h` to `__NR_syscalls`.

 - An arrow points from `./linux/arch/386/kernel/syscall_table.S` to the `sys_call_table` structure.

<http://www.ibm.com/developerworks/library/l-1-system-calls/>

- 1 Open file:
`/root/rpmbuild/BUILD/kernel-2.6.31/linux-2.6.31.i686/ipc/sem.c`
- 2 Implement your syscall function here.

Compilation

- 1 Compile new kernel

```
cd /root/rpmbuild/BUILD/kernel-2.6.3/linux-2.6.31.i686  
make
```

- 2 Copy to /boot partition

```
cp arch/x86/boot/bzImage /boot
```

- 3 Edit GRUB config file, create new entry for new kernel.

```
vim /boot/grub/grub.conf
```

- 4 reboot

Compilation

- 1 Compile new kernel

```
cd /root/rpmbuild/BUILD/kernel-2.6.3/linux-2.6.31.i686  
make
```

- 2 Copy to /boot partition

```
cp arch/x86/boot/bzImage /boot
```

- 3 Edit GRUB config file, create new entry for new kernel.

```
vim /boot/grub/grub.conf
```

- 4 reboot

Compilation

- 1 Compile new kernel

```
cd /root/rpmbuild/BUILD/kernel-2.6.3/linux-2.6.31.i686  
make
```

- 2 Copy to /boot partition

```
cp arch/x86/boot/bzImage /boot
```

- 3 Edit GRUB config file, create new entry for new kernel.

```
vim /boot/grub/grub.conf
```

- 4 `reboot`

Compilation

- 1 Compile new kernel

```
cd /root/rpmbuild/BUILD/kernel-2.6.3/linux-2.6.31.i686  
make
```

- 2 Copy to /boot partition

```
cp arch/x86/boot/bzImage /boot
```

- 3 Edit GRUB config file, create new entry for new kernel.

```
vim /boot/grub/grub.conf
```

- 4 reboot

Task:

- Implement a syscall function *sys_cnt_low* that accepts a C-string *S* (null-terminated) which returns number of characters from interval $[a - m]$ from string *S*.
- Implement a C program that calls *sys_cnt_low* and verifies its correctness on 1 sample input string.
- Max length of the string is 128 B.
- Hint: kernel space vs. user address space
 - `get_user`, `put_user`
 - `copy_from_user`, `copy_to_user`
 - `strncpy_from_user`, `strlen_from_user`

Linux cheatsheet

- Editors: vim, nano
- Vim cheatsheet: <http://vim.rtorr.com/>
- Vim tutorial: http://www.viemu.com/a_vi_vim_graphical_cheat_sheet_tutorial.html
- Bash (linux command line) cheatsheet:
<http://www.cheat-sheets.org/saved-copy/fwunixref.pdf>