MC504 - Sistemas Operacionais

Chamadas de Sistema

Islene Calciolari Garcia

Instituto de Computação - Unicamp

Primeiro Semestre de 2016

Sumário

Objetivos

- Configurar um ambiente para testes
- Teste inicial: `printk`
- Criar uma nova chamada de sistema

QEMU

QEMU is a generic and open source machine emulator and virtualizer.
When used as a machine emulator, QEMU can run OSes and programs made for one machine (e.g. an ARM board) on a different machine (e.g. your own PC). By using dynamic translation, it achieves very good performance.
When used as a virtualizer, QEMU achieves near native performances by executing the guest code directly on the host CPU. QEMU supports virtualization when executing under the Xen hypervisor or using the KVM kernel module in Linux. When using KVM, QEMU can virtualize x86, server and embedded PowerPC, and S390 guests.

Fonte: www.qemu.org

QEMU

- Imagem disponível no site do QEMU
  - Testing QEMU
  - kernel muito antigo: 2.6.20
- Uso: `$ qemu-system-i386 linux-0.2.img`

Como rodar um kernel (ainda não) alterado

QEMU

- Crie uma imagem de disco vazio com o comando
  `$ qemu-img create -f qcow disco.img 5G`
- qcow: QEMU copy on write
- Imagem terá no máximo 5G
- Este passo é fácil. ;-)

Como rodar um kernel (ainda não) alterado

QEMU

- Faça uma instalação a partir de um CD real, colocado na máquina hospedeira.
  `$ qemu-system-i386 -hda disco.img -cd-rom /dev/cdrom -boot d`
- ou a partir de um arquivo qualquer com uma imagem de CD:
  `$ qemu-system-i386 -hda disco.img -cd-rom image-cd.iso -boot d`
- A maioria das distribuições não funciona. :-(
- Debian parece ser uma exceção.
- Veja mais dicas no site Fedora: how to use QEMU

Como rodar um kernel (ainda não) alterado

QEMU

- Para rodar um kernel diferente do da imagem, faça:
  `$ qemu-system-i386 -hda disco.img -kernel bzImage -append ‘‘ro root=/dev/hda’’`
- Obtenha uma versão do kernel Linux a partir de kernel.org
- Execute `tar -xJvf linux.4.X.Y.xz`
- Obtenha um arquivo `.config` adequado: veja config-4.5.3
- `make -j 5 ARCH=i386`
- O kernel estará em `arch/x86/boot/bzImage`
- Tempo de compilação nesta máquina: cerca de 10 minutos

Primeiro printk

- Alternativa do kernel para o printf
- Saída será exibida na tela e/ou irá para um arquivo de log
- Comando `dmesg` mostra as mensagens
- Imagem `linux.0.2.img` não tem `dmesg`. :-(
- Podemos utilizar `mc504.img` criada por Glauber de Oliveira Costa para a turma de 2008
- Sugestão: alterar arquivo `init/calibrate.c`

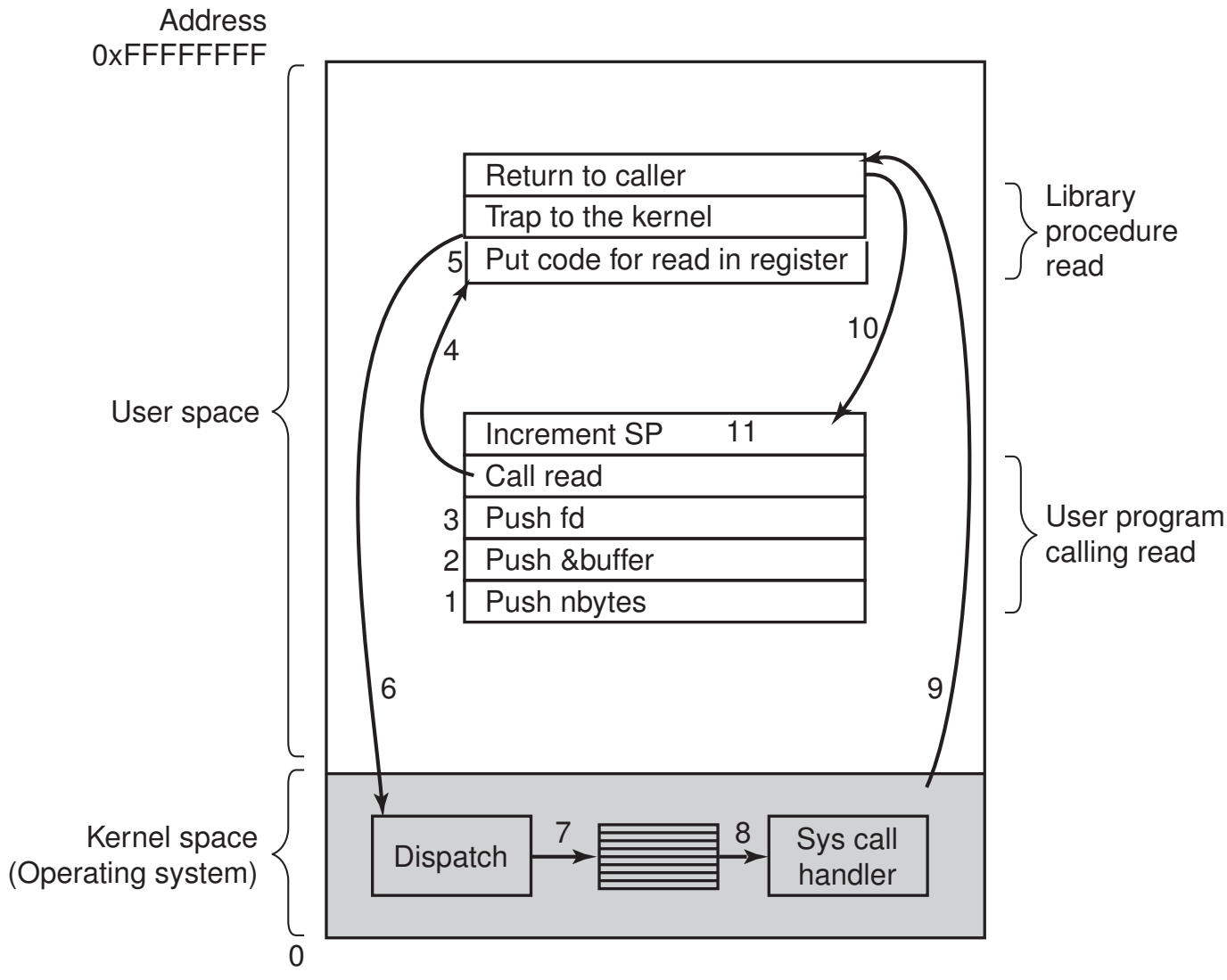
Espaço de endereçamento e kmalloc

- Verificar endereços das variáveis
- Como funciona o kmalloc/kfree?
- Sugestão: alterar arquivo `init/calibrate.c`

```
#include <linux/slab.h>
...
char *p = kmalloc(size, GPF_KERNEL);
```

Chamadas de Sistema

- Fronteira entre o espaço de usuário e o espaço do kernel (modo privilegiado)
- Tabela de chamadas
- Deslocamento via número da chamada



Tanenbaum: Figure 1-17

Process management

Call	Description
pid = fork( )	Create a child process identical to the parent
pid = waitpid(pid, &statloc, options)	Wait for a child to terminate
s = execve(name, argv, environp)	Replace a process' core image
exit(status)	Terminate process execution and return status

File management

Call	Description
fd = open(file, how, ...)	Open a file for reading, writing or both
s = close(fd)	Close an open file
n = read(fd, buffer, nbytes)	Read data from a file into a buffer
n = write(fd, buffer, nbytes)	Write data from a buffer into a file
position = lseek(fd, offset, whence)	Move the file pointer
s = stat(name, &buf)	Get a file's status information

Directory and file system management

Call	Description
s = mkdir(name, mode)	Create a new directory
s = rmdir(name)	Remove an empty directory
s = link(name1, name2)	Create a new entry, name2, pointing to name1
s = unlink(name)	Remove a directory entry
s = mount(special, name, flag)	Mount a file system
s = umount(special)	Unmount a file system

Miscellaneous

Call	Description
s = chdir(dirname)	Change the working directory
s = chmod(name, mode)	Change a file's protection bits
s = kill(pid, signal)	Send a signal to a process
seconds = time(&seconds)	Get the elapsed time since Jan. 1, 1970

Tanenbaum: Figure 1-18

Chamadas de Sistema

- Normalmente utilizadas via *wrapper functions*
- Outra alternativa: `syscall`
- Veja exemplo: `myfutex`

Adição de uma nova entrada na tabela

- Incluir linha em `arch/x86/syscalls/syscall_32.tlb`
- Incluir declaração de função em `include/linux/syscalls.h`
- Incluir código em `arch/x86/kernel/`
- Alterar o `Makefile`
- Recompilar o `kernel`

Como testar a nova chamada

- Criar um arquivo exemplo: `ex-mycall.c`
- Compilar fora da imagem com
  - `$ gcc -m32 -static ex-mycall.c -o ex-mycall`
  - Opção -m32 nem sempre habilitada corretamente...
  - Tentar ccplusplus
- Deixar o executável visível como se fosse um disco
  - `$ qemu ... -hdb ex-mycall`
- Execute no sistema sendo emulado:
  - `$ cat /dev/hdb/ > ex-mycall`
  - `$ chmod +x > ex-mycall`
  - `$ ./ex-mycall`

Conclusão

- É relativamente fácil incluir uma nova funcionalidade no kernel via chamadas de sistema.
- Por que esta abordagem não é sempre a melhor?