Sistemas Operacionais 2

Programação: Chamadas de Sistema

Hélio Crestana Guardia
helio@dc.ufscar.br

Universidade Federal de São Carlos
Departamento de Computação

System Calls

- Todo **sistema operacional** provê **interface** para que programas solicitem **serviços** do núcleo.
- Variações de Unix oferecem funções bem definidas, chamadas ***system calls***, ou **chamadas de sistema**.
- Definição das chamadas feita em **C**, independentemente do mecanismo utilizado para solicitação dos serviços.
- Cada chamada de sistema normalmente tem uma função com o mesmo nome definida em **C**.
- Processo de usuário chama função usando a sequência de chamada normal da linguagem.
- Função faz chamada apropriada ao serviço do *kernel,* colocando valores nos **registradores do *hardware*** (e possivelmente também na **pilha**) e chamando instrução de **interrupção**, ou outra instrução de acesso ao SO

GNU C Library

- *Any Unix-like operating system needs a C library: the library which defines the "system calls" and other basic facilities such as open, malloc, printf, exit...*
- *The GNU C library is used as the C library in the GNU system and most systems with the Linux kernel.*
- *The history of Unix and various standards determine much of the interface of the C library. In general the GNU C library supports the ISO C and POSIX standards. We also try to support the features of popular Unix variants (including BSD and System V) when those do not conflict with the standards. Different compatibility modes (selectable when you compile an application) allow the peaceful coexistence of compatibility support for different varieties of Unix.*

http://www.gnu.org/software/libc

Unix Standards

- POSIX: *Portable Operating System Interface*
- Família de padrões desenvolvidos pelo IEEE e adotados por ISO (*International Organization for Standardization*) e IED (*International Electrotechnical Commission*): ISO/IEC.
  - *1003.1: System Interface*
  - *1003.1a: System Interface Extensions*
  - *1003.2: Shell and Utilities*
  - *1003.2a: Shell and Utilities - Tools & User Port. Ext.*
  - *1003.2b: Additional Utilities*
  - *1003.1b: Realtime*
  - *1003.1c: Threads*
  - *1003.1d: Realtime Extensions*
  - *1003.1j: Advanced Realtime Extensions*
  - *1003.1e/.2c: Security*
  - ...

System Interface Table: http://www.unix.org/version3/inttables.pdf
Common utilities: http://www.unix.org/version3/apis/cu.html

Chamadas de Sistema no Linux

- *$ man syscalls*
- *The system call is the fundamental interface between an application and the Linux kernel.*
- *System calls are generally not invoked directly, but rather via wrapper functions in glibc.*
- *Often the glibc wrapper function is quite thin, doing little work before invoking the system call, and then setting erro appropriately after the system call has returned.*

syscalls

- _llseek(2), _newselect(2), _sysctl(2), accept(2), access(2), acct(2), adjtimex(2), afs_syscall, alarm(2), bdflush(2), bind(2), break, brk(2), cacheflush(2), capget(2), capset(2), chdir(2), chmod(2), chown(2), chown32, chroot(2), clone(2), close(2), connect(2), creat(2), create_module(2), delete_module(2), dup(2), dup2(2), execve(2), exit(2), fchdir(2), fchmod(2), fchown(2), fchown32, fcntl(2), fcntl64, fdata-sync(2), flock(2), fork(2), fstat(2), fstat64, fstatfs(2), fsync(2), ftime, ftruncate(2), ftruncate64, get_kernel_syms(2), getcwd(2), getdents(2), getdents64, getegid(2), getegid32, geteuid(2), geteuid32, getgid(2), getgid32, getgroups(2), getgroups32, getitimer(2), getpagesize(2), getpeername(2), getpmsg, getpgid(2), getpgrp(2), getpid(2), getppid(2), getpriority(2), getresgid(2), getresgid32, getresuid(2), getresuid32, getrlimit(2), getrusage(2), getsid(2), getsockname(2), getsockopt(2), gettid, gettimeofday(2), getuid(2), getuid32, gtty, idle, init_module(2), ioctl(2), ioperm(2), iopl(2), ipc(2), kill(2), lchown(2), lchown32, link(2), listen(2), lock, lseek(2), lstat(2), lstat64, madvise(2), mincore(2), mkdir(2), mknod(2), mlock(2), mlockall(2), mmap(2), modify_ldt(2), mount(2), mprotect(2), mpx, mremap(2), msync(2), munlock(2), munlockall(2), munmap(2), nanosleep(2), nfsservctl(2), nice(2), oldfstat, oldlstat, oldolduname, oldstat, oldumount, olduname, open(2), pause(2), personality(2), phys, pipe(2), pivot_root(2), poll(2), prctl(2), pread(2), prof, profil, ptrace(2), putpmsg, pwrite(2), query_module(2), quotactl(2), read(2), readahead, readdir(2), readlink(2), readv(2), reboot(2), recv(2), recvfrom(2), recvmsg(2), rename(2), rmdir(2), rt_sigaction, rt_sigpending, rt_sigprocmask, rt_sigqueueinfo, rt_sigreturn, rt_sigsuspend, rt_sigtimedwait, sched_get_priority_max(2), sched_get_priority_min(2), sched_getparam(2), sched_getscheduler(2), sched_rr_get_interval(2), sched_setparam(2), sched_setscheduler(2), sched_yield(2), security, select(2), sendfile(2), send(2), sendmsg(2), sendto(2), setdomainname(2), setfsgid(2), setfsgid32, setfsuid(2), setfsuid32, setgid(2), setgid32, setgroups(2), setgroups32, sethostname(2), setitimer(2), setpgid(2), setpriority(2), setregid(2), setregid32, setresgid(2), setresgid32, setresuid(2), setresuid32, setreuid(2), setreuid32, setrlimit(2), setsid(2), setsockopt(2), settimeofday(2), setuid(2), setuid32, setup(2), sgetmask(2), shutdown(2), sigaction(2), sigaltstack(2), signal(2), sigpending(2), sigprocmask(2), sigreturn(2), sigsuspend(2), socket(2), socketcall(2), socketpair(2), ssetmask(2), stat(2), stat64, statfs(2), stime(2), stty, swapoff(2), swapon(2), symlink(2), sync(2), sysfs(2), sysinfo(2), syslog(2), time(2), times(2), truncate(2), truncate64, ulimit, umask(2), umount(2), uname(2), unlink(2), uselib(2), ustat(2), utime(2), vfork(2), vhangup(2), vm86(2), vm86old, wait4(2), waitpid(2), write(2), writev(2).

Programação em C

- Compilador: cc / gcc // *man gcc*
- Uso:

gcc prog.c [-o prog] [bib.o] [-Iinc_dir] [-Llib_dir] [-llink_bib]

- Ex:

gcc prog.c	// comp., link e gera. código: **a.out**
gcc prog.c -o prog	// gera: **prog**
gcc -c prog.c	// comp. apenas, gera **prog.o**
gcc prog.c -lm	// a.out, incluindo **libm**

Programação em C

Comandos do pré-processador (cpp)

- *man cpp // C Preprocessor*

#include // #include<unistd.h> (*)
#define // #define _OPENMP
#ifdef / #endif // #ifdef _OPENMP

#if defined (_OPENMP) && defined (VERBOSE)
...
#else
...
#endif

(*) Localização dos *includes*:
#include <xxx.h> // biblioteca padrão, em ***/usr/include***
#include "xxx.h" // diretório local, ou especificado com -I

Programação em C

Link e bibliotecas

-lbib // inclui biblioteca *lib**bib**.a* (*)
-Llib_dir // especifica caminho de busca

(*) Localização das bibliotecas:
Link estático: ***/usr/lib*** (também bibs dinâmicas adicionais)
Link dinâmico: ***/lib, /usr/lib, (/etc/ld.so.conf)***

export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:....

Compilação com **link dinâmico** (*shared objects*) - *default*:
gcc ... -static // previne ligação com bibliotecas dinâmicas
gcc ... -shared // produz objeto compartilhado que pode ser "*linkado*" com
// outros objetos para formar um programa.

ldd: exibe dependências de bibliotecas dinâmicas // *man ldd*
ldconfig: configura ajustes para links dinâmicos // *man ldconfig*

make

- Utilitário para **manutenção** de grupos de programas
- Usa informações de um arquivo de configuração (***GNUmakefile***, ***makefile***, ***Makefile***, ou outro especificado)
- Ações baseadas em **regras** e nas **datas** de última modificação dos arquivos envolvidos.
- Executa comandos no ***makefile*** para atualizar um ou mais ***targets***, tipicamente programas

Makefile

DEFINIÇÕES
target: componentes
<Tab> regra
Ex:

```
DIR=/usr/local
OPT=-Wall
PROGS=prog teste

all: prog teste

clean:
	rm -f $(PROGS)

install:
	cp $(PROGS) $(DIR)/bin

prog:	bib.o prog.c
	gcc $(OPT) prog.c bib.o -o prog

bib.o:	bib.c
	gcc $(OPT) -c bib.c

teste:	teste.c
	gcc $(OPT) teste.c -o teste
```

Depuração (*debug*)

gcc -W__ // define nível de *warnings*. Ex: *-Wall*
gcc -g // gera programa instrumentado para depuração
gcc -g*level*

man gcc
/ Warning Options
/ Debugging Options

gdb *prog*
help *// !*
*break _nome_função_*
run / start
nexti / next
kill *// !*
quit

Chamadas de Sistema

- Retorno:

0: sucesso // quase sempre!

-1: erro

- Tratamento de erro:

#include <errno.h>

errno

perror( )

strerror( )