Programming Assignments
Information Session
Tao Gong
Pthread library

• Thread:
  • pthread_create, pthread_join, pthread_exit;

• Mutex
  • pthread_mutex_lock; pthread_mutex_unlock;

• CV
  • pthread_cond_wait; pthread_cond_signal; pthread_cond_broadcast;
Programming Assignment 2: part 1

• Mutual exclusive accessing CurrentID
  • Create 1 mutex
  • Lock & unlock, create a critical section
  • In CS, check CurrentID and proceed
  • Create 6 threads and pass parameters (thread ids)

• Takeaway:
  • Thread creation; Critical section
  • Feel the inefficiency caused by starvation
Programming Assignment 2: part 2

• Mutual exclusive accessing queue, signaling to avoid starvation
  • Create 1 mutex
  • Lock & unlock, create a critical section (to access a queue element)
  • Check queue element. *Give up lock and wait if cannot proceed* (pthread_cond_wait, full for producer and empty for consumer)
  • Produce or consume. Notify waiting threads if they can proceed now (pthread_cond_broadcast, produced consumed the first element)
  • Create 2+3 threads

• Takeaway:
  • Thread signaling
  • Starvation avoidances
  • (OPT) Threads cancellation techniques
Programming Assignment 3: os161 kernel

• Part A: warming up
• Part B, C, D: create 3 system calls and their implementation (in kernel)
• Part E, F: create user applications to test created system calls

• _exit() <- a stub in place
• printint()
• reversestring()

• Later two need return values
How "reboot" syscall is called

- reboot() is declared in include/unistd.h
- reboot() is called by userapps, (so they must include unistd)
- reboot() is defined by a macro SYSCALL according to the callno.h by the script:
  - callno-parse.sh file loads callno.h, read the segment between /*CALLBEGIN*/ /*CALLEND*/
  - remove the leading SYS_
  - SYSCALL macro (syscalls-mips.S ) creates functions and actually point to __syscall(SYS_callname)
  - e.g.: #define SYS_reboot 8
  - callno-parse.sh and SYSCALL macro WILL create reboot() and is actually calling __syscall(SYS_reboot,)
  - This process is automated by make (or make depends I can't remember exactly)
  - (so you only need to define it in the callno.h segment with leading SYS_)
- os161 captured __syscall and now you are at mips_syscall() (arch/mips/mips/syscall.c), kernel context
- the switch-case statement routes the execution to sys_reboot(), also passes parameters
- sys_reboot() is declared in kern/include/syscall.h
- sys_reboot() is defined in kern/main/main.c
- kern/main/main.c is included in the kernel by kern/conf/conf.kern
Demo: creating system calls: printstring

- System call function
  - kern/include/kern/callno.h
  - System call “printstring” is automatically created:
    - printstring() declare and defined in userspace (stub)
    - SYS_printstring macro in kernel space

- Kernel space (implementation)
  - kern/arch/mips/mips/syscall.c
  - Switch cases
  - If create other C file, include in kern.conf

- User application (test application)
  - testbin/ copy a sample
  - Modify .c, Makefile, depend.mk
  - Modify ../Makefile

- Improvement
  - Parameter passing, return values, separate file.
Compile and execute

•Kernel config
  • at kern/conf: ./config ASST0

•Kernel code
  • at compile/ASST0: make depend; make; make install

•User code
  • make

•Execute OS
  • sys161 kernel-ASST0
Q & A
printf & kprintf?

• printf:
  • User space API
  • Standard C I/O library
  • Format the output and use system call to display it
    • Remember, kernel has your "screen", user application does not have direct control
    • The implementation is provided by kernel

• kprintf:
  • Kernel space API
  • Not a standard, but a common API
  • Same formatting, but directly display it at kernel
    • Where to display? It depends. It may be the screen, or kernel messaging buffer

• In os161, the display systemcall is not implemented, if you use printf, you will see “unknown system call number”
User space functions & kernel space functions

• Can I call user space functions from kernel space?
  • E.g., printint() and sys_printint() at kernel

• TLDR: You only call kernel space function at kernel side
How to modify Makefiles and depend.mk

• testbin/testprintstring/Makefile and depend.mk
  • Change every “add” to “testprintstring”

• testbin/Makefile
  • Copy the “add” line and modify “add” to “testprintstring”
Location of the files

- kern/include/kern/callno.h
- kern/arch/mips/mips/syscall.c
- testbin/testprintstring/*