OS 2019

MIT ;)

https://pdos.csail.mit.edu/6.828/2019

Systemove volania Entry/Exit

Struktura prednasky

- O co pojde
- RISC-V
- Ukazka prechodu user → kernel

Tema

Tema

- Ako funguje prechod do/z jadra pri systemovych volaniach

Tema

- Ako funguje prechod do/z jadra pri systemovych volaniach
- Ponorime sa do jadra OS

write(fd, buf, n)

write(fd, buf, n)

- Mozeme pouzit v uzivatelskom priestore funkciu (z pohladu jazyka C) na [za|vy] volanie (uskutocnenie) sluzby jadra?

write(fd, buf, n)

- Mozeme pouzit v uzivatelskom priestore funkciu (z pohladu jazyka C) na [za|vy] volanie (uskutocnenie) sluzby jadra?

- Bolo by to rychle a efektivne
- Taktiez flexibilne na prenos komplexnejsich udajovych typov
- A islo by o programatorom znamy mechanizmus

write(fd, buf, n)

- Zial, nemozeme pouzit takyto mechanizmus!

write(fd, buf, n)

- Zial, nemozeme pouzit takyto mechanizmus!
- Dovodom je IZOLACIA procesov

write(fd, buf, n)

- Zial, nemozeme pouzit takyto mechanizmus!

- Dovodom je IZOLACIA procesov
- Izolacia ovplyvnuje vacsinu navrhu jadra OS

Co to je izolacia

Co to je izolacia

- Vynutena separacia z dovodu zapuzdrenia dosledkov zlyhani

Co to je izolacia

- Vynutena separacia z dovodu zapuzdrenia dosledkov zlyhani
- Predmetom izolacie je zvycajne proces

Co to je izolacia

- Vynutena separacia z dovodu zapuzdrenia dosledkov zlyhani
- Predmetom izolacie je zvycajne proces
- Zabranuje
  - Procesu X modifikovat/citat udaje o procese Y (napr. Modifikovat tabulku deskriptorov, r/w pristupy do pamate…)
  - Procesu miesat sa do uloh a sluzieb OS

Hlavne nastroje izolacie

Hlavne nastroje izolacie

- Virtualny adresny priestor procesu

Hlavne nastroje izolacie

- Virtualny adresny priestor procesu

- Privilegovany rezim cinnosti CPU (proces nemoze robit I/O operacie a menit dolezite systemove registre CPU)

Hlavne nastroje izolacie

- Virtualny adresny priestor procesu
- Privilegovany rezim cinnosti CPU (proces nemoze robit I/O operacie a menit dolezite systemove registre CPU)
- Vynutena kontrola toku riadenia pomocou systemovych volani

Hlavne nastroje izolacie

- Virtualny adresny priestor procesu
- Privilegovany rezim cinnosti CPU (proces nemoze robit I/O operacie a menit dolezite systemove registre CPU)
- Vynutena kontrola toku riadenia pomocou systemovych volani; **prechod user → kernel!!!**

Privilegovany rezim CPU

Privilegovany rezim CPU

- RISC-V CPU 2 rezimy: supervisor, user

Privilegovany rezim CPU

- RISC-V CPU 2 rezimy: supervisor, user
- Kernel (jadro OS) bezi v rezime supervisor
- Bezny uzivatelsky program v rezime user

Privilegovany rezim CPU

- RISC-V CPU 2 rezimy: supervisor, user
- Kernel (jadro OS) bezi v rezime supervisor
- Bezny uzivatelsky program v rezime user
- Co moze supervisor oproti user naviac

Privilegovany rezim CPU

- RISC-V CPU 2 rezimy: supervisor, user
- Kernel (jadro OS) bezi v rezime supervisor
- Bezny uzivatelsky program v rezime user
- Co moze supervisor oproti user naviac:
  - I/O operacie (pristup k zariadeniam)
  - Nastavovanie adresneho priestoru (virtualna pamat)
  - R/W pristup k specialnym registrom CPU

Privilegovany rezim CPU

- RISC-V CPU 2 rezimy: supervisor, user
- Kernel (jadro OS) bezi v rezime supervisor
- Bezny uzivatelsky program v rezime user
- Co moze supervisor oproti user naviac:
  - I/O operacie (pristup k zariadeniam)
  - Nastavovanie adresneho priestoru (virtualna pamat)
  - R/W pristup k specialnym registrom CPU
- Temer kazdy CPU vyuziva tento princip

write(2, “$ “, 2)

write(2, “$ “, 2)

- Co sa deje pri takomto systemovom volani (shell vypisuje prompt)

write(2, “$ “, 2)

- Co sa deje pri takomto systemovom volani (shell vypisuje prompt)
  - user/sh.c vo fnc getcmd() fprintf(2, “$ “);

write(2, “$ “, 2)

- Co sa deje pri takomto systemovom volani (shell vypisuje prompt)
  - user/sh.c vo fnc getcmd() fprintf(2, “$ “);
  - user/printf.c fprintf()→ putc()→ write(fd, &c, 1);

write(2, “$ “, 2)

- Co sa deje pri takomto systemovom volani (shell vypisuje prompt)
  - user/sh.c vo fnc getcmd() fprintf(2, “$ “);
  - user/printf.c fprintf()→ putc()→ write(fd, &c, 1);
  - user/usys.S write

write(2, “$ “, 2)

- Co sa deje pri takomto systemovom volani (shell vypisuje prompt)
  - user/sh.c vo fnc getcmd() fprintf(2, “$ “);
  - user/printf.c fprintf()→ putc()→ write(fd, &c, 1);
  - user/usys.S write:
    - li a7, SYS_write
    - ecall
    - ret

write(2, “$ “, 2)

- Co sa deje pri takomto systemovom volani (shell vypisuje prompt)
  - user/sh.c vo fnc getcmd() fprintf(2, “$ “);
  - user/printf.c fprintf()→ putc()→ write(fd, &c, 1);
  - user/usys.S write:
    - li a7, SYS_write
    - ecall
    - ret
- Instrukcia **ECALL** !!!

Tok riadenia

Tok riadenia

write()

Tok riadenia

write()

trampoline.S / uservec

Tok riadenia

write()

trampoline.S / uservec

trap.c / usertrap()

Tok riadenia

write()

trampoline.S / uservec

trap.c / usertrap()

syscall.c / syscall()

Tok riadenia

write()

trampoline.S / uservec

trap.c / usertrap()

syscall.c / syscall()

sysfile.c / sys_write()

Tok riadenia

write()

trampoline.S / uservec

trap.c / usertrap()

syscall.c / syscall()

sysfile.c / sys_write()

trap.c / usertrapret()

Tok riadenia

write()

trampoline.S / uservec

trap.c / usertrap()

syscall.c / syscall()

sysfile.c / sys_write()

trap.c / usertrapret()

Tok riadenia

write()

trampoline.S / uservec

trap.c / usertrap()

syscall.c / syscall()

sysfile.c / sys_write()

trap.c / usertrapret()

trampoline.S / userret

Tok riadenia

write()

trampoline.S / uservec

trap.c / usertrap()

syscall.c / syscall()

sysfile.c / sys_write()

trap.c / usertrapret()

trampoline.S / userret

write()

Pamatovy priestor

- Obrazok na tabulu

- Uzivatel, Kernel
- Trampolina uplne hore
- Pod nou
  - Uzivatel trapframe – co to je?
  - Kernel zasobnik pre kazde jadro CPU

- kernel/memlayout.h

TF

TF

- Sluzi na uchovavanie stavu CPU pri prechode user→ kernel a naopak

TF

- Sluzi na uchovavanie stavu CPU pri prechode user→ kernel a naopak
- Vid kernel/proc.h

TF

- Sluzi na uchovavanie stavu CPU pri prechode user→ kernel a naopak
- Vid kernel/proc.h
  - Kernel page table
  - Kernel stack pointer
  - Address of usertrap() fnc in kernel
  - User pc CPU register
  - User 32 CPU registers

Niekolko RISC-V registrov

Niekolko RISC-V registrov

- Iba niekolko najdolezitejsich
- Dalsie vid kapitola 10 v https://github.com/Lingrui98/RISC-V-book/blob/master/rvbook.pdf

Niekolko RISC-V registrov

- stvec – supervisor trap-vector base addr reg
  - addr v jadre, kam skace ‘ecall’; addr trampoliny

Niekolko RISC-V registrov

- stvec – supervisor trap-vector base addr reg
  - addr v jadre, kam skace ‘ecall’; addr trampoliny
- sepc – supervisor exceptional instruction pc
  - Ecall vyvola vynimku; v tomto reg sa uchova adresa instrukcie, ktora vyvolala vynimku; v nasom pripade tu bude user pc (ecall je instrukcia v uzivatelskom programe)

Niekolko RISC-V registrov

- stvec – supervisor trap-vector base addr reg
  - addr v jadre, kam skace ‘ecall’; addr trampoliny
- sepc – supervisor exceptional instruction pc
  - Ecall vyvola vynimku; v tomto reg sa uchova adresa instrukcie, ktora vyvolala vynimku; v nasom pripade tu bude user pc (ecall je instrukcia v uzivatelskom programe)
- scause – supervisor cause
  - Ulozeny kod priciny vyvolanej vynimky; v pripade xv6 tam bude hodnota 8 (system call)

Niekolko RISC-V registrov

- sscratch – supervisor scratch
  - Jedno slovo (word) udajov na docasne pouzitie; v pripade xv6 tam je adresa trapframe

Niekolko RISC-V registrov

- sscratch – supervisor scratch
  - Jedno slovo (word) udajov na docasne pouzitie; v pripade xv6 tam je adresa trapframe
- satp – supervisor address translation and protection (riadi strankovanie)
  - Aktualna tabulka stranok

Ukazka user → kernel

Ukazka user → kernel

- Systemove volanie cez entry/exit je ovela komplexnejsie ako obycajne volanie fnc...

Ukazka user → kernel

- Systemove volanie cez entry/exit je ovela komplexnejsie ako obycajne volanie fnc…
- Takato komplikovana vec je v dosledku poziadavky izolacie procesov

Ukazka user → kernel

- Systemove volanie cez entry/exit je ovela komplexnejsie ako obycajne volanie fnc…
- Takato komplikovana vec je v dosledku poziadavky izolacie procesov
- Natiska sa legitimna otazka: neda sa to nejako jednoduchsie?

Ukazka user → kernel

- Systemove volanie cez entry/exit je ovela komplexnejsie ako obycajne volanie fnc…
- Takato komplikovana vec je v dosledku poziadavky izolacie procesov
- Natiska sa legitimna otazka: neda sa to nejako jednoduchsie?

- Odpoved… treba hladat ;)

Domace citanie

- Chapter 2: Operating system organization

Buduca prednaska

- Adresne priestory
- Tabulky stranok
- Virtualna pamat