VYSOKE UCENI TECHNICKE V BRNE
FAKULTA INFORMACNICH TECHNOLOGII

IOS - Operacni systemy

poznamky z prednasek

2019/2020

Corse & US Layout

Obsah

1

Prvni prednaska: Uvod do predmetu, prehled operacnich systemu, zakladni pojmy, jadro operacniho systemu a jejich typy, historie vyvoje operacnich systemu, prehled technickeho vybaveni, klasifikace pocitacu, operacnich systemu, hlavni smery ve vyvoji operacniho systemu.

1.1 Uvod, prehled operacnich systemu

Operacni system je vyznamnou casti vypocetnich systemu, ty zahrnuji:

- hardware,
- operacni system,
- uzivatelske aplikacni programy,
- uzivatele.

Prehled nekterych OS:

- GNU/Linux
  - GNU/Debian - Ubuntu
  - Red Hat - RHEL, Fedora, Cent OS
  - SuSE
  - Gentoo, Arch Linux, Slackware (= nejstarsi live distribuce linuxu)
- BSD
  - FreeBSD, OpenBSD
- GNU
  - zn. GNU Is Not Unix
- MS Windows
- Mac OS X
  - jadro XNU = X is Not Unix
- Android, iOS
- Minix
  - pouziva intel ve svych cipech

1.2 Zakladni pojmy

Operacni system je program (resp. kolekce programu), ktera vytvari spojujici mezivrstvu mezi hardware operacniho systemu a uzivateli a jejich uziv. aplik. programy. OS dale spotrebovava zdroje, jako jsou pamet nebo cas CPU. (tldr: sw, spojujici hardware, uzivatele a programy)

Cile OS:

- maximalni vyuziti zdroju pocitace - drahe pocitace, levnejsi pracovni sila (drive)
- jednoduchost pouziti pocitacu - levne pc, draha pracovni sila (dnes prevazuje)

Zakladni role OS:

- spravce prostredku
  - pamet, procesor, periferie
  - dovoluje sdilet prostredky efektivne a bezpecne
- tvurce prostredi pro uzivatele a jejich aplikacni programy
  - vytvareni abstrakci, virtualnich objektu (resp. poskytuje standardni rozhrani, ktere zjednodusuje prenositelnost aplikaci a zauceni uzivatelu)
  - abstrakce jsou napr.: proces, program, soubor
  - problemy abstrakci jsou mensi efektivita a nepristupne nektere nizkourovnove operace

OS zahrnuje:

- jadro (kernel),
- systemove knihovny a utility (= systemove aplikacni programy),
- textove (shell) ci graficke uzivatelske rozhrani (X Window).

Presna definice, co vse OS zahrnuje neexistuje. Ruzne firmy a komunity to chapou ruzne. (GNU to chape napr. jako projekt svobodneho OS, zahrnujici jadro, utility, GUI, TUI, vyvojove prostredky a knihovny, ...)

definice:

proces je aktivita rizena programem
program je predpis, navod na nejakou cinnost zakodovany vhodnym zpusobem
soubor je kolekce zaznamu (obvykle Byte) slouzici primarne jako zakladni jednotka pro ukladani dat na vnejsich pametovych mediich
adresar je kolekce souboru

1.3 Jadro operacniho systemu

Jedna se o nejnizsi a nejzakladnejsi cast OS. Zavadi se jako prvni a bezi po celou dobu behu pocitacoveho systemu (tzv. reaktivni system, spis nez transformacni). Navazuje primo na hardware (pripadne virtualizovany HW) a pro uzivatele a uziv. aplik. zcela zapouzdruje.

Bezi v privilegovanem rezimu:

- je mozne menit obsah registru hw, je mozne zadavat prikazy hw (neni mozne v uzivatelskem rezimu)
- musi byt podporovano v hardware

Jadro (obecne) zajistuje:

- zakladni spravu prostredku a tvorbu zakladniho prostredi jak pro uzivatele tak pro zbytek OS
- zahrnuje vsechny operace, kdy je potreba primo komunikovat s hardware (prepinani kontextu - jadro, plaovani procesu - nekdy v jadru, nekdy mimo, zavedeni stranky z disku, ..)
- sluzby pro zbytek OS a uzivatele, nektere zajistuje automaticky
- nektere sluby nejsou poskytovany automaticky, musi si o ne zadat, nazyvame to volani sluzeb, tzv. *system-call* (= systemova volani), ktere musi byt implemenovana uzitim specializovanych instrukci (intel: sw preruseni, syscall, sysenter)

Rozlisujeme dva typy rozhrani OS:

- *kernel interface* (nebo taky: ABI, Kernel ABI) - prime volani jadra pomoci specializovanych instrukci
- *library interface* - rozhrani vyssi urovne (napr. C knihovny), typicke sluzby jsou napr. printf z C - volaji se funkce ze systemovych knihoven, mohou ale nemusi vest na volani sluzeb jadra (bezne aplikace pracuji s timto rozhranim)

definice:

transformacni system je system, ktery dostane nejaky vstup, zpracuje ho a udela nejaky vystup (prekladac) - pokud se zacykli = chyba

reaktivni system se spusti a do (teoreticky) nekonecna reaguje na podnety uzivatele (spust proces - spusti proces) - pokud prestane pracovat = chyba

prepinani kontextu je situace, kdy na CPU bezi proces, ten chci pozastavit a nechat bezet jiny proces

instrukce syscall a sysenter - jakmile aplikace (bezi v uziv. rezimu) zavola takovou instrukci, dojde ke kontrolovanemu prepnuti do rezimu jadra, provede se sluzba, a pote se prepne zpet

ABI = Application Binary Interface

1.4 Typy jader OS

Monoliticka jadra

- vysokourovnove komplexni rozhrani s radou sluzeb, abstrakci, ktere mohou pouzivat vyssi vrstvy OS
- vsechny subsystemy jsou implementovany v privilegovanem rezimu, rezimu jadra, a zahrnuji napr. spravu pameti, planovani, meziprocesovou komunikaci, souborove systemy, ..
- vyhody: vysoka efektivita diky provazanosti
- nevyhody: mala flexibilita pri praci s jadrem (ve filesystemu je chyba, chci zmenit jen implementaci filesystemu za novou verzi a vse ostatni nechat - nelze, je nutne cely system zastavit a znovu nastarovat, nelze menit nic za behu)

Monoliticka jadra s modularni strukturou

- vylepseni koncepce monolitickych jader
- umoznuje zavadet/odstranovat subsystemu jadra v podobe tzv. modulu za behu
- vyhody: neni nutne cely system zastavovat a znovu bootovat pro vymenu jednoho modulu, vyssi bezpecnost - zavedou se jen moduly, ktere se budou pouzivat
- pouzivane v napr. FreeBSD, Linux

Mikrojadra

- snaha minimalizovat rozsah jadra a rozsah jeho sluzeb
- nabizi jednoduche rozhrani, maly pocet abstrakci, sluzeb, typicky nabizi nejzakladnejsi spravu CPU. I/O zarizeni, pameti, ..
- vetsina sluzeb nabizenych monolitickymi jadry (ovladace, vyznamne casti spravy pameti, planovani) je implemenovana mimo jadro v tzv. serverech (nebezi v privilegovanem rezimu).
- vyhody: flexibilita (vice soucaasne bezicich implementaci ruznych sluzeb, dynamicke spousteni, zastavovani..), zabezpeceni (chyba v serveru / utok na ne neznamena ovladnuti celeho OS, ale jen daneho serveru)
- nevyhody: vyrazne vyssi rezie

Generace mikrojader

- 1. generace - napr. Mach
- 2. generace - napr. L4, mensi rezie nez 1. gen
- 3. generace - napr. seL4 nebo ProvenCore, duraz na zabezpeceni, navrh s ohledem na moznost formalni verifikace

Hybridni jadra

- ”neco mezi mikrojadry a monolitickymi jadry”
- jadra zalozena na mikrojadrech, rozsirena o kod, ktery by mohl byt implementovan ve forme serveru, je ale za ucelem mensi rezie tesneji provazan s mikrojadrem a bezi v jeho rezimu
- pouzivane v napr. Mac OS X (Mach + BSD), Windows NT (a vyssi), ...

definice:

servery (v oblasti mikrojader) jsou procesy
formalni verifikaci rozumime overeni urcitych vlastnosti systemu s platnosti matematickeho dukazu

linux prikazy:

lsmod - vypise aktualne zavedene moduly jadra
rmmod - maze moduly jadra
modprobe - zavadeni modulu do jadra

1.5 Historie vyvoje OS

definice:

preruseni je elektricky signal, ktery jde od periferie po sbernici k procesoru, na CPU vyvola obsluhu preruseni - mechanismus umoznujici rozbehnout operaci na periferii a o tu periferii se nestarat (periferie pote oznami konec operace) (podrobne se tomu venuje oddil 2.5)

multitasking je soucasny beh vice aplikaci na jednom procesoru (muze byt s preemtivnim nebo nepreemtivnim planovanim)

nepreemtivni planovani zn. ze uloha, kt. aktualne bezi na CPU muze byt od CPU "odstavena"pouze tehdy, kdyz nejak zakomunikuje s jadrem (= pozada o sluzbu jadra, napr. periferni operace), dokonce lze pouzit specializovane sluzby pro prepnuti kontextu (proces se dobrovolne vzda CPU, tzv. yield sluzby) - vyhoda: snadna implementace, nevyhoda: pokud se proces zacykli (chyba), cely system se zablokuje (porad bezi 1 uloha)

preemtivni planovani - proces muze byt odstaven od CPU bez nutnosti komunikace s jadrem, napr. pomoci preruseni (jakehokoli typu)

1.6 Prehled technickeho vybaveni

Procesor (CPU):

- radic, ALU, registry (IP, SP), instrukce, ..

Pamet:

- adresa
- hiearchie pameti (cache, RAM, disky, ... - bank pameti muze byt vice)
  - pameti se lisi spotrebou, kapacitou, rychlosti, cenou za jednotku
  - na vrcholu hiearchie jsou registry (nejrychlejsi, nejvyssi cena za jednotku, mala kapacita)
  - cache (vyrovnavaci pameti, ruznych urovni, L1 = level 1, L2, L3, ..)
  - primarni pamet RAM
  - sekundarni pameti - disky
  - vyrovnavaci pameti disku
  - tercialni pameti (zalohy - nejnizsi cena za jednotku, nejpomalejsi, nejvetsi kapacita)

Periferie:

- disk (HDD, SDD,..), klavesnice, monitor (I/O porty, preruseni, DMA)

Sbernice:

- propojuji jednotlive komponenty
- na vrcholu hiearchie jsou sbernice propojujici CPU a pamet (FSB - Front Side Bus, HyperTransport QPI - Quick Path Interconnect)
- diskove sbernice (SATA/ATA, SCSI/SAS, USB)

- dalsi sbernice (NVLink - pripojovani nVidia GPU, PCI - rozsirujici karty ci disky, CAPI - IBM Tauer CPU, propojovani CPU a akceleratoru)

definice:

I/O porty = vstup-vystupni porty, predstavuji pametove oddeleny prostor od adresoveho prostoru bezne pameti, s temito adresami se komunikuje specialnimi instrukcemi (intel: inout)

pametove mapovane I/O je cast adresoveho prostoru bezne pameti neni pouzita pro praci s pameti, ale adresy jsou presmerovane do HW (neco co zapisu na danou adresu nebude v pameti ale v nejakem registru HW)

DMA zn. Direct Memory Access, souvisi s nezavislou cinnosti periferii - periferie mohou primo komunikovat s hardware (radic disku si sam z adresy pameti nacte data a pres sbernice je prenasi na disk, nebo naopak)

1.7 Klasifikace pocitacu

Dle ucelu:

- univerzalni,
- specializovane
  - vestavene (palubni pc, spotrebni elektronika, ..)
  - aplikacne orientovane (rizeni db, sitove servery, ..)
  - vyvojove (zkouseni novych technologii)

Podle vykonnosti:

- vestavene pc, tablety, mobily, ..
- osobni pocitace (PC) a pracovni stanice (workstation) - dnes se nerozlisuje
- servery
- strediskove pocitace (mainframe) - vyrabi IBM, ladene na obrovsky I/O vykon a vysokou spolehlivost
- superpocitace - ladene na surovy vypocetni vykon (vedecke vypocty, simulace)

1.8 Klasifikace OS

Podle ucelu:

- univerzalni (UNIX, Linux, Windows, ..)
- specializovane (real-time - RT-Linux, databaze, web - z/VSE, mobilni - iOS, Android)

Podle poctu uzivatelu:

- jednouzivatelske (CP/M, MS-DOS,..)
- viceuzivatelske (UNIX, Windows, ..)

Podle poctu soucasne bezicich uloh:

- jednoulohove
- viceulohove (multitasking, ne/preemptivni)

definice:

soft real-time - doporuceni aby se akce vykonavaly v realnem case
hard real-time - akce se musi vykonavat v urcitem case

1.9 Implementace OS

OS se obtizne programuji a ladi, protoze to jsou velke programove systemy, paralelni a asynchronni systemy, systemy zavisle na technickem vybaveni.

Dusledky:

- setrvacnost pri implementaci (snaha nemenit kod, ktery pracuje spolehlive)
- pouzivani technik pro minimalizaci vyskytu chyb (inspekce zdrojoveho kodu, rozsahle testovani, podpora vyvoje technik formalni verifikace)

definice:

paralelni system zn. ze zde bezi vice aktivit soucasne
paralelni asynchronni systemy - procesy se prepinaji v okamzicich, ktere nelze dopredu presne predpovedet

1.10 Hlavni smery ve vyvoji OS

- neustale vylepsovani architektur (snizovani rezii jader,)
- bezpecnost, spolehlivost
- podpora stale vetsiho poctu procesoru, vice jader
- virtualizace
- distribuovane zpracovani (cloudy, kontejnery, Internet of Things)
- OS tabletu, mobilu, vestavenych systemu, ...
- vyvoj novych technik navrhu a implementace OS (podpora formalni verifikace)

definice:

bezpecnost zn., ze system je odolny vuci vnejsim utokum
spolehlivost zn., ze system ”nespadne sam od sebe”

2

Druha prednaska: Unix - uvod: historie UNIXu (nezkousi se), priciny uspechu UNIXu, varianty UNIXu, zakladni koncepty, struktura jadra, komunikace s jadrem - hardwarova preruseni. Prehled programovani v UNIXu: nastroje programatora,

2.1 Priciny uspechu UNIXu

- viceprocesovy, viceuzivatelsky,
- napsan v C - prenositelny,
- zpocatku (a pozdeji) siren ve zdrojovem tvaru,
- ”mechanism, not policy”,
- ”fun to hack”,
- jednoduche uzivatelske rozhrani (terminal),
- skladani slozitejsich programu z jednodussich (tvoreni aplikaci typu filtr),
- hierarchicky system souboru,
- konzistenti rozhrani perifernich zarizeni

definice:

”mechanism, not policy” zn. snaha oddelit casti aplikaci (napr. GUI - oddelit zakladni rutiny pro vykreslovani grafiky od politik, tzn. koncove nastavby - barvy oken, umisteni tlacitek, .. -¿ systematicke rozdeleni vede k lepsim optimalizacim a ladenim algoritmu a zaroven rychlym zmenam politik)

”fun to hack” zn., lide se na vyvoji podili, protoze je to bavi (nejen protoze jsou za to placeni)

aplikace typu filtr - jednoduche otevrene aplikace, na vstupu maji textovy dokument v otevrene podobe, vstup zpracuji a na vystupu opet otevreny dokument (zadne binarni, zakodovane)

2.2 Varianty UNIXu

Hlavni vetve OS UNIXoveho typu:

- UNIX System V (puvodni system z AT&T),
- BSD UNIX (FreeBSD, NetBSD, ..),
- firemni varianty (AIX, Solaris, ..)
- Linux

Sousisejici normy:

- XPG - X/OPEN, SVR4 - AT&T,SUN, OSF/1, Single UNIX Specification,
- POSIX - IEEE standard,
- Single UNIX Specification v3/v4 - shell, utility (CLI), API

definice:

POSIX je striktni podmnozina Single UNIX Specification, je to standard definujici zakladni textove prikazove rozhrani OS + API

2.3 Zakladni koncepty

Jsou dve zakladni koncepce (abstrakce) UNIXu: **procesy** a **soubory**.

Procesy mezi sebou komunikuji pomoci ruznych mechanismu meziprocesove komunikace - IPC (Inter-Process Communication) - roury, signaly, semafory, sdilena pamet, sockets, zpravy, streams, .. a pro komunikaci pouzivaji nejake I/O rozhrani (read, write, close, ..)

definice:

procesy jsou abstrakci probihajici nejake aktivity (viz 1.2)
soubory jsou abstrakci dat (viz 1.2)

2.4 Struktura jadra UNIXu

Zakladni podsystemy jsou sprava souboru a sprava procesu.
Popis:

- Na hornim okraji jadra (smerem k uzivatelum, aplikacim) je vrstva implementujici rozhrani volani sluzeb, prostrednictvim ktere jadro prebira zadosti o sluzby od aplikaci. Jadro kontroluje zda ten, kdo o sluzbu zada ji muze volat, zda jsou parametry validni a rozhrani predava pozadavek dal do jadra.
- Aplikace mohou s jadrem komunikovat primo, nicmene nejcasteji komunikuji s jadrem pres knihovny. (viz. kernel & library interface)
- Na druhem okraji (tesne nad HW) je vrstva abstrakce hadrware.
- Mezi spravou souboru a hardware se nachazi ovladace, pote vrstva vyrovnavacich pameti, ktere souborove systemy pouzivaji ke zrychleni prace s relativne pomalymi disky (HDD, SSD - oproti RAM pomale) - OS se snazi vyhnout opakovanemu cteni stejnych dat, proto si v jednom okamziku nacte vic dat nez uzivatel zada, ulozi si data do vyrovnavaci pameti (pri dostatku pameti) a data nacita odtud. (napr. C knihovny jsou pouzivane kazdym druhym programem - jsou v pameti temer porad).

definice:

ovladace jsou programy slouzici k rizeni (zadavani prikazu, prebirani stavovych informaci, reseni mimoradnych stavu konkretnich periferii) - lze je (jako i prislusna zarizeni) rozdelit na znakova a blokova

znakova zarizeni jsou zarizeni komunikujici po jednotlivych znacich (klavesnice)

blokova zarizeni komunikuji po blocich (disk - sektory, resp. bloky)

komunikaci s jadrem rozumime nastavovani parametru hardware, vydavani prikazu hw, obsluhu ruznych stavu do kterych se hw dostava (a o kterych je CPU a jadro informovano prostrednictvim preruseni)

nastavovani parametru hw se deje pomoci I/O portu nebo pametove mapovanych operaci (viz. 1.x)

2.5 Komunikace s jadrem a hardwarova preruseni

Sluzby jadra jsou operace, jejich realizace je pro procesy zajistovana jadrem. Explictne je mozne o provedeni urcite sluzby zadat prostrednictvim system call (viz 1.3).

Priklady nekterych sluzeb jadra (systemova volani v UNIXu):

- open, close, read - otevre/zavre/cte soubor,
- write - zapisuje,
- kill - posle signal,
- fork - duplikuje proces,
- exec - prepise kod,
- exit - ukonci proces.

2.5.1 Hardwarove preruseni

- hardware interrupt je mechanismus, kterym HW zarizeni oznamuji jadru asynchronne vznik udalosti, ktere je zapotrebi obslouzit (dalsi mozna definice viz 1.5),
- zadosti o HW preruseni prichazi jako elektricke signaly (IRQ) do radice preruseni (APIC),
- procesor s radicem preruseni komunikuje pomoci I/O portu.

Prijem nebo obsluju HW preruseni lze zakazat:

- maskovanim preruseni,
- na CPU (instrukce CLI/STI na Intel/AMD - zakazou se vsechna krome NMI),
- ciste programve v jadre (preruseni se prijme, ale jadro si jen poznamena jeho prichod a neobsluhuje se)

NMI:

- non-maskable interrupt je HW preruseni, ktere nelze zamaskovat na radici ani zakazat na CPU,
- pouziva se pri kritickych chybach pameti, sbernice, .. (alternativne se pouziva pro ladeni / reseni uvaznuti v jadre ”NMI watchdog”)

Preruseni mohou vznikat i v CPU - jsou to synchronni preruseni, tzv. vyjimky (= exceptions):

- trap - po obsluze se pokracuje dalsi intrukci (breakpoint, overflow, ..)
- fault - po obsluze se znovu opakuje intrukce, ktera vyjimku vyvolala (vypadek stranky, deleni 0, ..)
- abort - dochazi k zavaznym problemum detekovanym CPU, neni jasne jak pokracovat - provedeni se ukonci (zanorene vyjimky typu fault, chyby HW detekovane CPU)

Mohou existovat i dalsi typy preruseni: (tato preruseni obsluhuje CPU zcela specifickym zpusobem (casto mimo vliv jadra, napr. na Intel/AMD))

- Interprocessor interrupt (IPI)
  - meziprocesorove preruseni
  - pouziva se pro preposilani preruseni z jednoho CPU na druhy nebo pro spravu cache (kazdy CPU ma svoji cache, do nich mohou mit CPU nacteny stejne adresy z pameti - pokud dojde ke zmenam v pameti, musi CPU informovat ostatni CPU o zmene)
- System management Interrput (SMI)
  - preruseni typu sprava systemu
  - muze byt vyvolano HW i SW ve zvlastnich situacich
  - pokud se takove preruseni vyvola, tak se dostane ke slovu firmware, ktery provadi obsluhu ruznych chybovych stavu (prehrati, vybita baterie, ..)
  - v ramci SMI nebezi bezne aplikace ani jadro, nesmi obsluha SMI bezet prilis dlouho (system se muze dostat do nekonzistentniho stavu)

2.5.2 Zakazovani preruseni

Proc preruseni zakazovat?

- v ramci obsluhy jednoho preruseni muze nastat dalsi preruseni,
- napr. na CPU bezi vypocet, neco nastane na disku, disk posle preruseni, to dojde k CPU a jadro zacne preruseni obsluhovat, v ten moment se neco stane na klavesnici a prijde dalsi preruseni,
- pote dale v ramci obsluhy muze jadro upravovat ruzne sve interni struktury, ktere mohou byt v nekonzistentnim stavu (napr. zretezene seznamy procesu [ukazatele], ruzne si je projuje, nez je stihne propojit, prije dalsi proces a muze sahnout do pameti kam nema),
- proto obsluha preruseni musi byt synchronizovana a v pripade, ze se v ramci preruseni provadi nejaka kriticka operace je nutne vyloucit ostatni (vsechna) preruseni

2.5.3 Pristupy k zakazovani preruseni

Pokud vsak zakazu (nejaka/vsechna) preruseni, abych se mohl venovat obsluze jednoho a budu ho obsluhovat prilis dlouho, system se muze dostat do nekonzistentniho stavu (jako u SMI). Pouzivaji se proto dva pristupy:

- je snaha zakazovat jen preruseni s nizsimi prioritami,
- rozdelit obsluhu preruseni do vice casti (urovni).

Obsluha preruseni je casto delena na dve urovne:

- 1. uroven:
  - ma byt co nejkratsi,
  - v ramci obsluhy preruseni se zakomunikuje nezbytnym zpusobem s HW (prevzani dat z/do HW, vydani prikazu HW, ..) a naplanuje se beh 2. urovne,
  - nelze pouzit bezne synchronizacni prostredky (protoze napr. CPU bezi nejaky vypocet, prijde preruseni z disku, jadro zacne resit 1. uroven obsluhy, nicmene obsluha != proces)
- 2. uroven:
  - dokoncuje obsluhu preruseni,
  - provadi se operace, kdy neni potreba komunikovat s hardware,
  - nemusi se zakazovat preruseni,
  - muze bezet v specialnich procesech (interrup threads ve FreeBSD nebo tasklety/softIRQ v Linuxu),
  - mohou se pouzit bezne synchronizacni prostredky

2.5.4 Ovladace zarizeni a preruseni

- pri inicializaci ovladace (v Linuxu je to typicky modul) nebo pri jeho prvnim pouziti se musi registrovat k obsluze urciteho IRQ,
- bud u nekterych zarizeni se pouzivaji (historicky) zafixovana cisla preruseni,
- nebo ovladace muze zjistit cislo preruseni tak, ze zakomunikuje s radicem sbernic, pokud to nefunguje,
- ovladac vyda prikaz zarizeni, ktere ma ovladat, aby zacalo vysilat nejaka preruseni (a ”poslouchala”sbernici, ”kdo se ozve”),
- pote se zaregistruje k obsluze prislusneho preruseni a hardware se pres tabulku preruseni ovladac ”dostane ke slovu”,
- vice zarizeni vsak muze pouzivat stejne cislo zadosti o preruseni
  - v takovem pripade jadro vytvori zretezeny seznam ovladacu, ktere maji zajem o dane preruseni
  - ovladace musi byt napsane tak, ze pokud jim dojde preruseni (o ktere maji zajem), tak musi zakomunikovat s tim zarizenim a zeptat se ho, zda opravdu to zarizeni poslalo dane preruseni
  - pokud ano - obslouzi se, pokud ne - preda se rizeni preruseni dalsimu ovladaci v seznamu

2.5.5 Priklad komunikace s jadrem

Synchronni komunikace je proces-jadro, asynchronni je hardware-jadro. Priklad:

- proces A zavola sluzbu read() a jadro ihned zacne volani obsluhovat (synchronni)
- nejprve se podiva do cache zda data, o ktera ma zajem proces A uz tam nejsou
- pokud ano, tak mu je rychle nakopiruje z cache na adresu, kterou pozaduje proces (bez komunikace s diskem)
- pokud data nejsou v cache, proces A bude pozastaven a jadro vyda prostrednictvim ovladacu disku prikaz k nacteni urciteho objemu dat, typicky vice nez zada uzivatel a nacita do vyrovnavaci pameti (ne na pozadovanou adresu)
- na procesoru dale bezi proces B, taky pozada o read(), zopakuje se to same co u A
- az disk dokonci operace jednoho z procesu (nemusi byt v poradi volani), disk posle preruseni na CPU
- jadro bude informovano, ze ma potrebna data pro proces A/B
- z cache nakopiruje pozadovana data na pozadovanou adresu
- pote se proces A/B probudi a bezi dal, to same se stane u dalsiho procesu

definice (pro 2.5.x):

asynchronni zn., bez prime-okamzite vazby na to co dela jadro nebo aplikace (tiskarna tiskne - operace nekdy skonci -¿ ale nikdy nevim dopredu kdy presne)

synchronni zn., ze CPU neco provede a ihned se zavola preruseni (napr. deleni 0)

IRQ = interrupt request

radic preruseni = interrput controller, hardwarova jednotka, ktera predava preruseni do CPU - registruje prichozi IRQ, ty se dle priorit predavaji do CPU (preruseni je mozne take zamaskovat - nepredavat dal do CPU) v podobe cisla preruseni, CPU se automaticky prepne do chraneneho rezimu a spusti obsluznou rutinu definovanou jadrem (preruseni 1 - provede xxx, 2 - xxx, ..)

APIC = Advanced Programmable Interrupt Controller - distribuovany system, kazdy CPU ma lokalni APIC, externi zarini mohou byt pripojena primo / pres I/O APIC

NMI watchdog - jadro si nadefinuje, ze casovac mu kazdych n casovych jednotek posle toto preruseni - pokud dojde v jadre k uvaznuti pri obsluze jineho preruseni a vsechna preruseni budou zakazana, toto se vzdy dostane do CPU (jadro se muze zotavit)

vypadek stranky zn., (pamet je rozdelena na casti, ktere mohou byt rozdeleny na disk) kdyz proces bude sahat do pameti a sahne na stranku, ktera v ni neni - detekuje se ze stranka tam neni -¿ poruseni ochrany pameti - jadro zkontruluje, zda proces nesaha kam by nemel, a pokud ne, tak mu stranku nahraje zpet do pameti a znovu se provede ta stejna instrukce

bezne synchronizacni prostredky jsou napr semafory nebo zamky a synchronizuji procesy

linux:

zakladni statistiky o obsluze preruseni jsou v */proc/interrupts*

2.6 Nastroje programatora UNIXu

X-Window system, vzdaleny pristup pres X-Window uzitecne prikazy na linuxu, ovladani vimu, apod. - vice viz. 2. prednaska IOS, u zkousky to nebyva.

3

3.1 Bash, shell, experimenty

4

4.1 Bash, shell, experimenty

Treti a ctvrta prednaska je venovana hlavne shellu, prochazi se prakticky ruzne prikazy a provadi se experimenty, apod. - lepsi je shlednout + na zkousce nic takoveho nebyva.

5

Pata prednaska: Sprava souboru: pevny disk,

5.1 Sprava souboru

TODO