

VYSOKÉ UCENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

IOS - Operační systémy
poznámky z přednášek

Obsah

1		2
1.1	Uvod, prehled operacnich systemu	2
1.2	Zakladni pojmy	3
1.3	Jadro operacniho systemu	4
1.4	Typy jader OS	5
1.5	Historie vyvoje OS	7
1.6	Prehled technickeho vybaveni	7
1.7	Klasifikace pocitacu	8
1.8	Klasifikace OS	9
1.9	Implementace OS	9
1.10	Hlavni smery ve vyvoji OS	10
2		11
2.1	Priciny uspechu UNIXu	11
2.2	Varianty UNIXu	12
2.3	Zakladni koncepty	12
2.4	Struktura jadra UNIXu	13
2.5	Komunikace s jadrem a hardwarova preruseni	14
2.5.1	Hardwarove preruseni	14
2.5.2	Zakazovani preruseni	15
2.5.3	Pristupy k zakazovani preruseni	16
2.5.4	Ovladace zarizeni a preruseni	16
2.5.5	Priklad komunikace s jadrem	17
2.6	Nastroje programatora UNIXu	17
3		18
3.1	Bash, shell, experimenty	18
4		18
4.1	Bash, shell, experimenty	18
5		19
5.1	Sprava souboru	19

1

První přednáška: Úvod do predmetu, prehľad operacnich systemu, zakladni pojmy, jadro operacniho systemu a jejich typy, historie vyvoje operacnich systemu, prehľad technickeho vybaveni, klasifikace pocitacu, operacnich systemu, hlavni smery ve vyvoji operacniho systemu.

1.1 Úvod, prehľad 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), která vytváří spojující mezivrstvu mezi hardware operacního systému a uživateli a jejich uživ. aplik. programy. OS dále spotřebovává zdroje, jako jsou paměť nebo čas CPU. (tldr: sw, spojující hardware, uživatele a programy)

Cile OS:

- maximalní využití zdroje počítače - drahé počítače, levnější pracovní síla (drive)
- jednoduchost použití počítače - levné pc, drahá pracovní síla (dnes převládá)

Zakladni role OS:

- správce prostředků
 - paměť, procesor, periférie
 - dovoluje sdílet prostředky efektivně a bezpečně
- tvůrce prostředí pro uživatele a jejich aplikací programy
 - vytváření abstrakcí, virtuálních objektů (resp. poskytuje standardní rozhraní, které zjednodušuje přenositelnost aplikací a zručení uživatele)
 - abstrakce jsou např.: proces, program, soubor
 - problémy abstrakcí jsou menší efektivita a nepřístupné některé nízkourovňové operace

OS zahrnuje:

- jádro (kernel),
- systémové knihovny a utility (= systémové aplikací programy),
- textové (shell) či grafické uživatelské rozhraní (X Window).

Přesná definice, co vše OS zahrnuje neexistuje. Různé firmy a komunity to chápou různě. (GNU to chápe např. jako projekt svobodného OS, zahrnující jádro, utility, GUI, TUI, vývojové prostředky a knihovny, ...)

definice:

proces je aktivita řízená programem

program je předpis, návod na nějakou činnost zakódovaný vhodným způsobem

soubor je kolekce záznamů (obvykle Byte) sloužící primárně jako základní jednotka pro ukládání dat na vnějších paměťových médiích

adresář je kolekce souborů

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 zapouzduje.

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, nektare zajistuje automaticky
- nektare 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 (spusti 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

Monolitická jadra

- vysokourovnove komplexni rozhrani s radou sluzeb, abstrakci, které mohou používat vyšší vrstvy OS
- všechny subsystemy jsou implementovány v privilegovaném režimu, režimu jádra, a zahrnují např. správu paměti, plánování, meziprocsovou komunikaci, souborové systémy, ..
- výhody: vysoká efektivita díky provazanosti
- nevýhody: malá flexibilita při práci s jádrem (ve filesystému je chyba, chci změnit jen implementaci filesystému za novou verzi a vše ostatní nechat - nelze, je nutné celý systém zastavit a znovu nastarovat, nelze měnit nic za běhu)

Monolitická jadra s modulární strukturou

- vylepšení koncepce monolitických jader
- umožňuje zavádět/odstraňovat subsystemy jádra v podobě tzv. modulu za běhu
- výhody: není nutné celý systém zastavovat a znovu bootovat pro výmenu jednoho modulu, vyšší bezpečnost - zavedou se jen moduly, které se budou používat
- používají se např. FreeBSD, Linux

Mikrojadra

- snaha minimalizovat rozsah jádra a rozsah jeho služeb
- nabízí jednoduché rozhraní, malý počet abstrakcí, služeb, typicky nabízí nejzákladnější správu CPU, I/O zařízení, paměti, ..
- většina služeb nabízených monolitickými jádry (ovládání, významné části správy paměti, plánování) je implementována mimo jádro v tzv. serverech (nebo v privilegovaném režimu).
- výhody: flexibilita (více současně bezcíh implementací různých služeb, dynamické spouštění, zastavování..), zabezpečení (chyba v serveru / útoku na něj neznamena ovládnutí celého OS, ale jen daného serveru)
- nevýhody: výrazně vyšší režie

Generace mikrojader

- 1. generace - např. Mach
- 2. generace - např. L4, menší režie než 1. gen
- 3. generace - např. seL4 nebo ProvenCore, důraz na zabezpečení, návrh s ohledem na možnost formální verifikace

Hybridni jadra

- "neco mezi mikrojadry a monolitickymi jadry"
- jadra zalozena na mikrojadrech, rozsirena o kod, který 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, který jde od periferie po sběrnici k procesoru, na CPU vyvolá obsluhu preruseni - mechanismus umožňující rozbehnout operaci na periférii a o tu periférii se nestarat (periferie poté oznámí konec operace) (podrobně se tomu věnuje oddíl 2.5)

multitasking je současný běh více aplikací na jednom procesoru (může být s preemtivním nebo nepreemtivním plánováním)

nepreemtivní plánování zn. že úloha, kt. aktuálně běží na CPU může být od CPU "odstavena" pouze tehdy, když nějak komunikuje s jádrem (= požádá o službu jádra, např. periferní operace), dokonce lze použít specializované služby pro přepnutí kontextu (proces se dobrovolně vzdá CPU, tzv. yield služby) - výhoda: snadná implementace, nevýhoda: pokud se proces zacyklí (chyba), celý systém se zablokuje (poradě běží 1 úloha)

preemtivní plánování - proces může být odstaven od CPU bez nutnosti komunikace s jádrem, např. pomocí preruseni (jakéhokoli typu)

1.6 Přehled technického vybavení

Procesor (CPU):

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

Paměť:

- adresa
- hierarchie pamětí (cache, RAM, disky, ... - bank pamětí může být více)
 - paměti se liší spotřebou, kapacitou, rychlostí, cenou za jednotku
 - na vrcholu hierarchie jsou registry (nejrychlejší, nejvyšší cena za jednotku, malá kapacita)
 - cache (vyrovnávací paměti, různých úrovní, L1 = level 1, L2, L3, ..)
 - primární paměť RAM
 - sekundární paměti - disky
 - vyrovnávací paměti disku
 - terciální paměti (zálohy - nejnižší cena za jednotku, nejpomalejší, největší kapacita)

Periferie:

- disk (HDD, SSD,...), klávesnice, monitor (I/O porty, preruseni, DMA)

Sběrnice:

- propojují jednotlivé komponenty
- na vrcholu hierarchie jsou sběrnice propojující CPU a paměť (FSB - Front Side Bus, HyperTransport QPI - Quick Path Interconnect)
- diskové sběrnice (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, spotrební 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 surový 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 uzivatelů:

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

Podle poctu soucasne bezicich uloh:

- jednoulahove
- viceulahove (multitasking, ne/preemptivni)

definice:

soft real-time - doporučení aby se akce vykonávaly v reálném čase

hard real-time - akce se musí vykonávat v určitém čase

1.9 Implementace OS

OS se obtížně programují a ladí, protože to jsou velké programové systémy, paralelní a asynchronní systémy, systémy závislé na technickém vybavení.

Důsledky:

- setrvačnost při implementaci (snaha nemění kód, který pracuje spolehlivě)
- používání technik pro minimalizaci výskytu chyb (inspekce zdrojového kódu, rozsáhlé testování, podpora vývoje technik formální verifikace)

definice:

paralelní systém zn. že zde běží více aktivit současně

paralelní asynchronní systémy - procesy se přepínají v okamžicích, které nelze dopředu přesně předpovědět

1.10 Hlavní směry ve vývoji OS

- neustále vylepšování architektur (snížení rezí jader,)
- bezpečnost, spolehlivost
- podpora stále většího počtu procesorů, více jader
- virtualizace
- distribuované zpracování (cloudy, kontejnery, Internet of Things)
- OS tabletu, mobilu, vestavěných systémů, ...
- vývoj nových technik návrhu a implementace OS (podpora formální verifikace)

definice:

bezpečnost zn., že systém je odolný vůči vnějším útokům

spolehlivost zn., že systém "nespadne sám od sebe"

2

Druha prednaska: Unix - uvod: historie UNIXu (nezkousi se), priciny uspechu UNIXu, varianty UNIXu, zakladni koncepty, struktura jadra, komunikace s jadem - 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,
- konzistentni 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 podobě, vstup zpracuji a na vystupu opet otevreny dokument (zadne binarni, zakodovane)

2.2 Varianty UNIXu

Hlavní větve OS UNIXového typu:

- UNIX System V (původní systém z AT&T),
- BSD UNIX (FreeBSD, NetBSD, ..),
- firemní varianty (AIX, Solaris, ..)
- Linux

Související 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 striktní podmnožina Single UNIX Specification, je to standard definující základní textové příkazy a rozhraní OS + API

2.3 Základní koncepty

Jsou dvě základní koncepce (abstrakce) UNIXu: **procesy** a **soubory**.

Procesy mezi sebou komunikují pomocí různých mechanismů meziprocesové komunikace - IPC (Inter-Process Communication) - roury, signály, semaforey, sdílená paměť, sockets, zprávy, streams, .. a pro komunikaci používají nějaké I/O rozhraní (read, write, close, ..)

definice:

procesy jsou abstrakcí probíhající nějaké aktivity (viz 1.2)

soubory jsou abstrakcí dat (viz 1.2)

2.4 Struktura jadra UNIXu

Zakladni podsystemy jsou sprava souboru a sprava procesu.

Popis:

- Na horním okraji jadra (směrem k uživateli, aplikacím) je vrstva implementující rozhraní volání služeb, prostřednictvím které jádro přebírá žádosti o služby od aplikací. Rozhraní kontroluje, zda ten, kdo o službu žádá, jí může volat, zda jsou parametry validní a rozhraní předává požadavek dál do jadra.
- Aplikace mohou s jádrem komunikovat přímo, nicméně nejčastěji komunikují s jádrem přes knihovny. (viz. 1.3)
- Na druhém okraji (těsně nad HW) je vrstva abstrakce hardware.
- Mezi správou souboru a hardware se nachází ovladač, poté vrstva vyrovnávací paměti, které souborové systémy používají ke zrychlení práce s relativně pomalými disky (HDD, SSD - oproti RAM pomale) - OS se snaží vyhnout opakovanému čtení stejných dat, proto si v jednom okamžiku načte více dat než uživatel žádá, uloží si data do vyrovnávací paměti (při dostatku paměti) a data načítá odtud. (např. C knihovny jsou používány každým druhým programem - jsou v paměti téměř pořád).

definice:

ovladace jsou programy sloužící k řízení (zadávaní příkazů, přebírání stavových informací, řešení mimořádných stavů konkrétních periférií) - lze je (jako i příslušná zařízení) rozdělit na znaková a bloková

znaková zařízení jsou zařízení komunikující po jednotlivých znacích (klávesnice)

bloková zařízení komunikují po blocích (disk - sektory, resp. bloky)

komunikaci s jádrem rozumíme nastavování parametrů hardware, vydávání příkazů HW, obsluhu různých stavů do kterých se HW dostává (a o kterých je CPU a jádro informováno prostřednictvím přerušení)

nastavování parametrů HW se děje pomocí I/O portu nebo pamětové mapovaných operací (viz 1.6)

2.5 Komunikace s jadem a hardwarova preruseni

Sluzby jadra jsou operace, jejich realizace je pro procesy zajistovana jadem. Explicitne 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 jadu asynchronne vznik udalosti, ktere je zapotrebi obslousit (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 obsluhu 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 poznamenava 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 instrukci (breakpoint, overflow, ..)
- fault - po obsluze se znovu opakuje instrukce, ktera vyjimku vyvolala (vypadek stranky, deleni 0, ..)
- abort - dochazi k zavaznym problemum detekovanim 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 Interrupt (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, který 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, které mohou byt v nekonzistentnim stavu (napr. zretezene seznamy procesu [ukazatele], ruzne si je projuje, nez je stihne propojit, prijde 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,
 - nemusí se zakazovat preruseni,
 - muze bezet v specialnich procesech (interrupt 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 prvni pouziti se musi registrovat k obsluze urcitého IRQ,
- bud u nekterych zarizeni se pouzivaji (historicky) zafixovana cisla preruseni,
- nebo ovladac 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 Příklad komunikace s jádrem

Synchronní komunikace je proces-jadro, asynchronní je hardware-jadro. Příklad:

- proces A zavola službu read() a jádro ihned začne volání obsluhovat (synchronní)
- nejprve se podívá do cache zda data, o která má zájem proces A už tam nejsou
- pokud ano, tak mu je rychle nakopíruje z cache na adresu, kterou požaduje proces (bez komunikace s diskem)
- pokud data nejsou v cache, proces A bude pozastaven a jádro vydá prostřednictvím ovladače disku příkaz k načtení určitého objemu dat, typicky více než žádá uživatel a načítá do vyrovnávací paměti (ne na požadovanou adresu)
- na procesoru daleko mezi procesy B, taky požádá o read(), zopakuje se to stejné jako u A
- až disk dokončí operaci jednoho z procesů (nemusí být v pořadí volání), disk pošle přerušeni na CPU
- jádro bude informováno, že má potřebná data pro proces A/B
- z cache nakopíruje požadovaná data na požadovanou adresu
- poté se proces A/B probudí a běží dál, to stejné se stane u dalšího procesu

definice (pro 2.5.x):

asynchronní zn., bez přímé okamžité vazby na to co dělá jádro nebo aplikace (tiskárna tiskne - operace někdy skončí - ale nikdy nevím dopředu kdy přesně)

synchronní zn., že CPU něco provede a ihned se zavola přerušeni (např. dělení 0)

IRQ = interrupt request

radic přerušeni = interrupt controller, hardwarová jednotka, která předává přerušeni do CPU - registruje příchod IRQ, ty se dle priorit předávají do CPU (přerušeni je možné také zamaskovat - nepředávat dál do CPU) v podobě čísla přerušeni, CPU se automaticky přepne do chráněného režimu a spustí obslužnou rutinu definovanou jádrem (přerušeni 1 - provede xxx, 2 - xxx, ..)

APIC = Advanced Programmable Interrupt Controller - distribuovaný systém, každý CPU má lokální APIC, externí zařízení mohou být připojena přímo / přes I/O APIC

NMI watchdog - jádro si nadefinuje, že časovač mu každých n časových jednotek pošle toto přerušeni - pokud přijde v jádře k uvažování při obsluze jiného přerušeni a všechna přerušeni budou zakázána, toto se vždy dostane do CPU (jádro se může zotavit)

vypadek stranky zn., (paměť je rozdělena na části, které mohou být rozděleny na disk) když proces bude sahát do paměti a sahne na stránku, která v ní není - detekuje se že stránka tam není - porušeni ochrany paměti - jádro zkontroluje, zda proces nesáhá kam by neměl, a pokud ne, tak mu stránku nahraje zpět do paměti a znovu se provede ta stejná instrukce

bezpečné synchronizační prostředky jsou např. semaforey nebo zamky a synchronizují procesy

linux:

základní statistiky o obsluze přerušeni jsou v */proc/interrupts*

2.6 Nástroje programátora UNIXu

X-Window systém, vzdálený přístup přes X-Window užitečné příkazy na linuxu, ovládání vimu, apod. - více viz. 2. přednáška IOS, u zkoušky to nebylo.

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

Patá prednaska: Sprava souboru: pevny disk,

5.1 Sprava souboru

TODO