Co je proces? - instance běžícího programu např. kompilování programu

**Má : Adresní prostor** **- textovou**(kod vykonávaný procesorem), **datovou**(proměnné a dyn. alokovaná paměť) a **zásobník**(stack)

**Taky: Obsahy registrů procesoru, otevřené soubory, I/O zařízení, obsahuje jedno nebo více vláken**

Proces je identifikovatelný **(PID)**

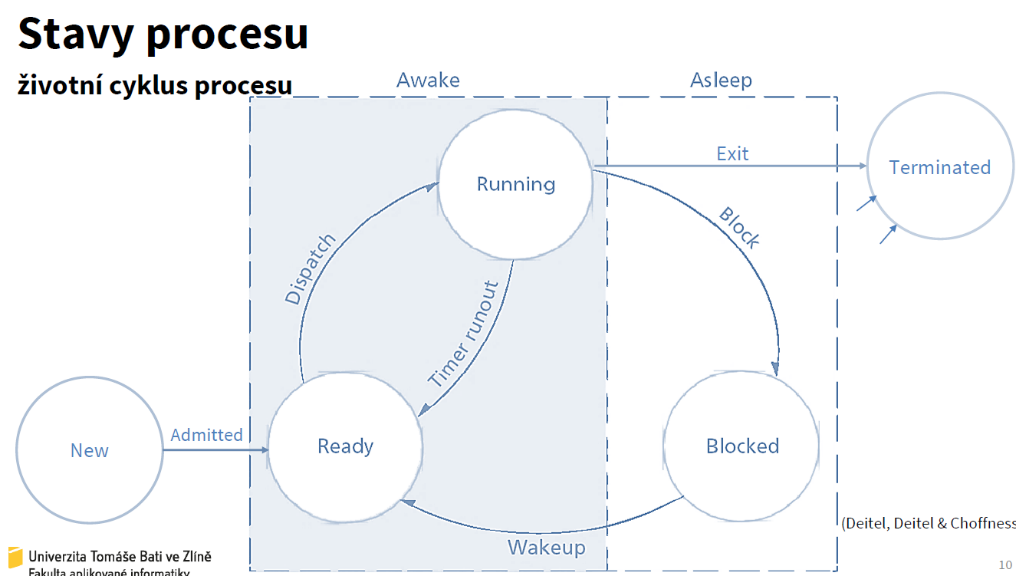
**V OS je kontejner vláken**

počítače vykonávají operace souběžně

**Job – task – program – proces – vlákno**

**Paralelismus x pseurdoparalelismus** – procesy můžou mít paralelní operace, jedno jádro cpu může pracovat s jedním vláknem

**Multitasking** – iluze běhu více procesů zároveň

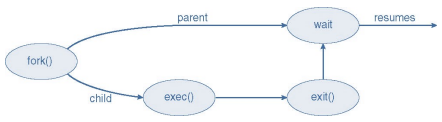


**Stavy procesu – new, running, ready, blocked(waiting), terminated**

**Správa procesů – create, destroy, dispatch, block, wake up, změna priority, IPC, synchronizace, suspend, resume**

**Vytvoření procesu –** pomocí system call, rodičovský a potomek(podprocesor)

**Vytvoření procesu v Linuxu – created by system call fork**

****

**Vytvoření procesu ve Windows –** funkce CreateProcess, nezávislý na rodiči, jestliže je úspěšná, tak vrací strukturu

**Implementace procesu** – OS spravuje tabulku procesů (Process table)

Každý proces má jeden záznam

**Záznam odkazuje na PCB** – deskriptor procesu, jedná se o datovou strukturu

**Deskriptor procesu (PCB)**

**Obsahuje informace o:** Správě procesů, procesoru, paměti, souborů, I/O…

**Info o správě procesů:** registry, stavy, ukazately, skupiny, čísla…

**Info o správě souborů:** akt. koř.adresář, euid, egid

**Info pro správu procesoru:** priorita procesu, čas

**Info pro správu paměti:** ukazatele, informace, euid, egid, stavy

**Střídání procesů:** **Multitasking** – iluzně vykonává více úkolů současně

Operační systém řídí multitasking tak, že přiděluje časový limit jednotlivým úlohám a střídá je, dokud nejsou všechny dokončeny.

Postup přepínání procesu:

1. Přeruší se běh procesu
2. Uloží se stav
3. Načte se stav jiného procesu

**Preemptivní** – používá časovač, zajišťuje čas. interval pro běh

**Kooperativní** – umožňuje, aby procesy byly dokočeny, procesy se vzdávají času CPU pro jiný proces. Může nastat problém, že proces spotřebuje celý čas CPU samostatně

Moderní OS používají Preemptivní

V přerušení se přerušuje aktuální proces, aby ho šlo později obnovit, uloží se stav (kontext)

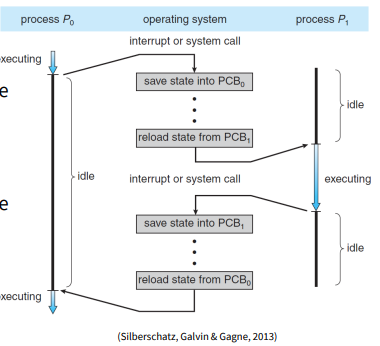
**Přepínání kontextu – skládá se z uložení a načítání (Con.S Con.L)**

Kontext se ukládá do PCB, každé přepnutí je režie

Má hardwarovou podporu, která je rychlá, ale běžný systém ji nepoužívá

Softwarové přepínání kontextu může být selektivní, není o dost pomalejší

Procesy neví, že se přepínají kontexty (transparentní)



**Přerušení kontextu** – moderní OS využívá interrupt driven

HW přerušení (Interrupt request) – signál zaslán do CPU

SW přerušení (výjimka – Exception, Trap (systemcall operace)

**Zpracování přerušení**

Obdržení přerušení procesor dokončí aktuální instrukci a potom zastaví proces – uloží se kontext do PCB

Procesor provede jednu z funkcí jádra pro zpracování přerušení – každý typ přerušení má index vektoru (interrupt vector)

Obslužný systém určuje reakci systému, po dokončení obsluhy obnoví přerušený nebo jiný proces a načte se kontext určitého procesu

Diagram

Description automatically generated

**Vlákna – odhlečený proces (lightweight process)**

Jsou plánována na CPU, neexistuje samo, patří do procesu, mohou být spravována OS KLT nebo ULT

Kontext vlákna je uložený v TCB

**Multithreading** – jeden proces více vláken

**Výhody vláken** – lepší spolupráce, výkon hardwaru, škálování, méně času na vytvoření a ukončení, přepínání, data se uloží v cache, efektivní na komunikaci

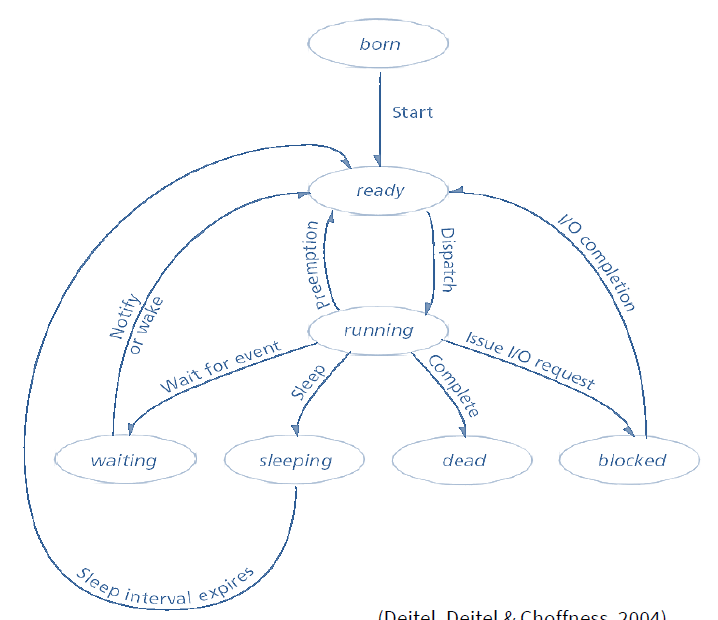
**Vláknové modely** – KLT – spravuje je kernel, ULT – spravuje je uživatel, kombinace

**Operace s vlákny** – Create, exit(terminate), suspend, resume, sleep, wake, cancel, join

**Stavy vláken** – Born, ready, running, waiting, sleeping, blocked, dead

**POSIX** – portable OS interface, má jednotné rozhraní a přenáší programy

**Pthreads** – knihovna, kde vlákna využívá POSIX vláknové API



**Souběh (Race Condition)**

Situace, kdy procesy přistupují ke sdíleným datům současně a manipulují

Konečnou hodnotu zjistíme podle toho, který proces skončí poslední

Souběhu zabráníme synchronizací souběžných procesů

**Problém kritické sekce**

Procesy lzou provádět souběžně, dají se kdykoliv přerušit, kód běží po částech

Může dojít k nekonzistenci dat, n procesů sdílí vyrovnávací pamět pevné vel.

Každý proces má část kódu “kritická sekce”, ve kterém přistupuje ke sdílené paměti

Problémem je, že potřebujeme zajistit, když jeden proces vykonává kritickou sekci, aby nebyl žádný jiný proces v kritické sekci

Procesy používají a modifikují sdílená data

Operace zápisu musí být pouze vzájemně i s operacemi čtení

Operace čtení mohou být souběžné, pro zabezpeční se dat se používají zámky

**Řešení kritické sekce**

Vzájemné vyloučení(pokud proces se nachází v kritické sekci, další procesy taky nesmí být v kritické sekci), progres, konečné čekání(musí existovat omezení procesů)

**Předpoklady**

Nesmí být dva procesy v kritické sekci, žádný proces nečeká věčně do vstupu a mimo kritické sekci neblokuje jiné procesy

**Aktivní čekání**

Nepřetržité testování proměnné, dokud se neobjeví hodnota¨

Používá ho spinlock, který je zámek k aktivnímu čekání

**Softwarové x Hardwarové řešení synchronizace**

**U softwarového** je striktní střídání procesů turn, přerušuje se požadavek, který je nepřípustný závislosti na rychlosti

Pokud je proces rychlejší, tak čeká než další projde kritickou sekcí a nastaví turn

**U hardwarové** je přístup k zamykacím proměnným atomický

Jednoprocesorové systémy můžou vypnout přerušení, pouze na kernel úrovni

Obsahuje speciální atomické instrukce

**Mutex(Mutual exclusion) Vzájemné vyloučení**

Chrání kritickou sekci získáním zámku acquire a uvolnuje release(), pracuje na principu booleanu (true, false)

Obsahuje pouze jeden úkol, používá se u Pthread synchronizace

**Semafor**

Je zobecněný mutex, přistupuje se díky dvou atomických operací wait() down() nebo signal() up()

**Rozdíly mezi semaforem a mutexem**

Mutex je uzamykací mechanismus používaný k synchronizaci přístupu ke zdroji, pouze jen jeden úkol (vlákno nebo proces) může získat mutex, více úkolů má přístup k jednom prostředku, jen jednotlivě

Semafor je signalizační mechanismus- signál, “jsem hotový můžeš pokračovat”, hodnotu semaforu se mění úlohou pracující se zdrojem, umožnuje více úkolům přistupovat k určitému počtu zdroje

**Spinlock**

Je obecný semafor (čítač), který používá aktivní čekání, blokování a přepínání je časově náročnější než ztráta času s krátkodobým aktivním čekáním

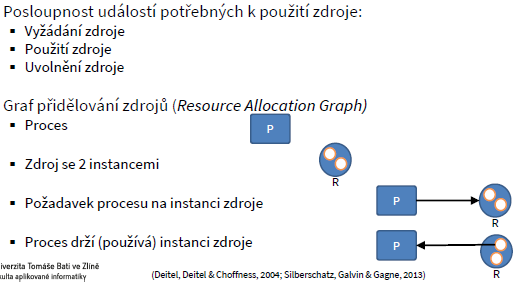
Používá se ve víceprocesorových systémech pro implementaci krátkých kritických sekcí, uvnitř kernelu

**Monitory**

Vysokoúrovnová abstrakce, která poskytuje pohodlný a účinný mechanismus pro synchronizaci procesů, má speciální konstrukci program.jazyka, současně může být na monitoru aktivní pouze jeden proces

**Negativní důsledky synchronizace**

**Uváznutí(Deadlock) je jedna z negativních důsledků synchronizací**

****

**Jednoduché uváznutí –** systém uváznul, protože procesy drží prostředek požadovaným jiným procesem a žádný proces nemůže uvolnit prostředek, který drží

Proces nebo vlákno čeká na událsot, která se nestane, skupina procesů je zablokována, pokud procesy ve skupině čekají na událost, který může způsobit jiný proces ve skupině

**Řešení uváznutí – Deadlock Prevention, D. Avoidance, D. Detection & D.Recovery, Ignoring Deadlock**

Prevence uváznutí - zajistí, že uváznutí nenastane

Vyhýbat se uváznutí – zajistit, že se nikdy nedojde k uváznutí

Detekce uváznutí a obnova – uváznutí může nastat a zajistí se obnova stavu

Ignorování hrozby

**Vyhladovění(starvation)**

**Aktivní zablokování(livelock)**

**Inverze priorit(priority inversion)**

**Virtuální paměť**

Představuje iluzi větší fyzické paměti, OS musí ukládat části virtuální paměti pro každý proces mimo hlavní pamět

Virtuální adresní prostor má logické zobrazení, jak je proces uložen v paměti. Má možnost sdílení kódu a dat.

Virtuální paměť a stránkování jsou funkce operačního systému, slouží k řízení a optimalizaci využití RAM

**Virtuální paměť** je funkce, stará se o alokaci paměti pro aplikace, které potřebují více paměti než je na počítači dostupné. Aplikace nevidí skutečnou paměť, ale virtuální paměť, který OS vytvořil na pevném disku. Jednotka správy paměti (MMU) musí logicky mapovat na fyzické adresy. Odděluje LAP od FAPChart

Description automatically generated

**Stránkování** je technika, kterou OS používá k optimalizaci využítí RAM. Proces, který potřebuje více paměti, než je k dispozici v RAM, se rozdělí na menší části (stránky) a ukládají se na pevném disku. Stránky se pak načítají a uvolňují podle potřeby.

1. Stránkování při spuštění – celý program je vložen do paměti,
2. Stránkování na žádost (Demand Paging) – načítá v případě výpadku stránky,
3. Předstránkování (Prepaging) – nahrává stránku pro použití
4. Čištění(Pre-cleaning) – změněné rámce jsou ukládány na disk, kdy není přetížený
5. Kopírování při zápisu (COW) – při vytvoření procesu(fork) se nekopírují žádné stránky, modifikací datové stránky nastane chyba, která vytvoří kopii stránky a umožní modifikaci

Diagram

Description automatically generated

**Princip** je, že OS ukládá a načítá stránky z pevného disku do RAM podle potřeb aplikací a stavu paměti. To umožňuje aplikacím pracovat s větším množstvím dat než je k dispozici v RAM, aniž by došlo k zastavení či selhání systému.

**EXT – souborový systém (extended file system)**

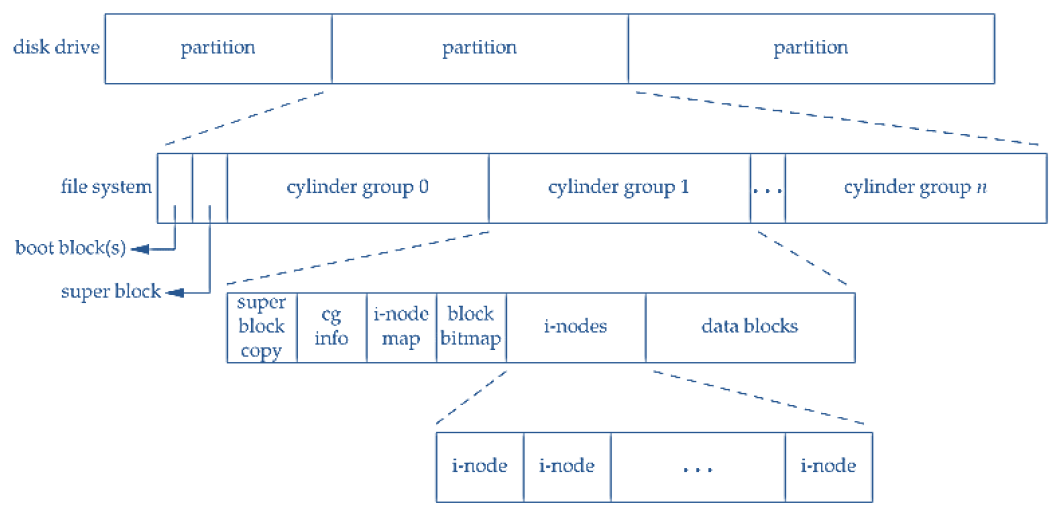
Je rodina souborových systémů, které umožňují ukládat a organizovat soubory na pevném disku PC.

Každý systém má jeden superblock, který se nachází na začátku přidělenému úložišti, obsahuje všechny informace o FS např.(stav připojení, ukazatel tabulky i-uzlů, velikost bloku, rozsah adres bloku), kopie superblocku jsou na několika dalších místech

Má min. jednu tabulku, která popisuje soubory, položky se nazývají i-uzly, obsahuje atributy souboru a mapu, kde jsou datové bloky souboru na disku

Jak funguje ext? - Ext se skládá z superblocku, i-uzlového stromu a blokového stromu. Superblock má informace o celém soub. sys., jako velikost disku, počet volných bloků atd. I-uzlový strom ukládá informace o souborech, blokový strom pak umožňuje organizaci bloků na disku k lehkému hledávání a přístupu k soub.

**Struktura ext**

****

Má redundantní kopii superbloku, tabulky deskriptorů skupiny, bitmapu bloků, i-uzlů, tabulka i-uzlů, datové bloky

Objekty v soub. syst. se představují i-uzlem, struktura i-uzlu má ukazatele na bloky soub. syst. (má všechna metadata o objektu kromě názvu)

Ext2, ext3, ext4

**FAT (File Allocation Table)**

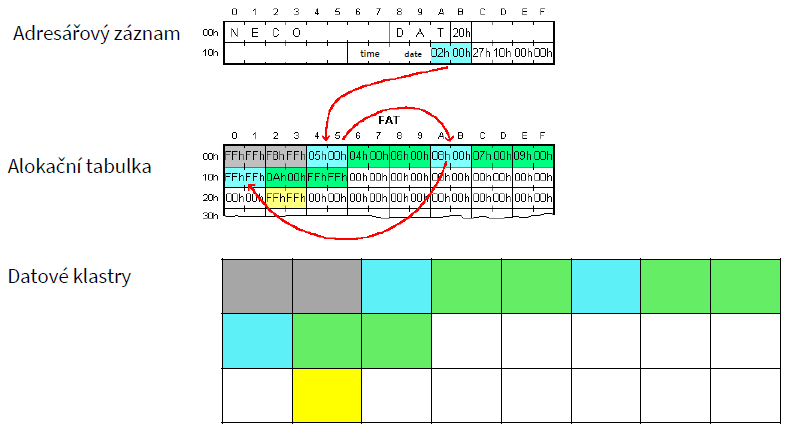
Příkladem je FAT12 – >16MiB, FAT16 - 16MiB-2GiB, FAT32 – 512MB-2TB a ExFAT(nástupce FAT) – teoreticky 128PiB

Je to jednoduchý souborový systém původně pro malé disky a jednoduché struktury složek, alokační tabulku a kořenovou složku má fixně uloženou, naformátovaný svazek je alokován po klastrech(cluster)

Klastr je nejmenší místo na disku k uložení souboru(alokační jednotka), větší soubor používá více klastrů, poslední klastr není plně využit (interní fragmentace) – ztracené množství prostrou na disku(vel. clust./2\*poč. souborů) velikost klastru je určena při formátování oddílu

**Struktura**

1. Vyhrazená oblast - Boot sektor oddíl (blok parametrů), obsahuje informace, které FS používá pro přístup ke svazku
2. Oblast FAT – dvě kopie alokačních tabulek pro soubory.
3. Kořenový adresář – popisuje soubory a složky v koř. adresáři oddílu, rozdíl s ostatními složkami je, že koř. složka je na určeném místě disku
4. Klastry – obsahuje soubory a adresáře



**Princip FAT** je založen na použití tabulky k řízení a alokaci prostoru na pevném disku, tabulka obsahuje informace o tom, která část disku je volná a která jsou použitá pro uložení souborů. Soubory jsou uloženy jako řetězce bloků na disku, kde bloky mají identifikační číslo a soubory vlastní řetězec bloků, který přiděluje prostor pro uložení dat. Informace jsou potom uloženy ve FAT tabulce. Při čtení či zápisu souboru OS využije informace ve FAT tabulce, kde určí, které části disku je potřeba pro čtení či zápis souboru. Pokud není místo, OS vyhledá volný blok na disku a přidělí ho souboru, aktualizuje FAT tabulku a uloží informace o nových blocích do ní.