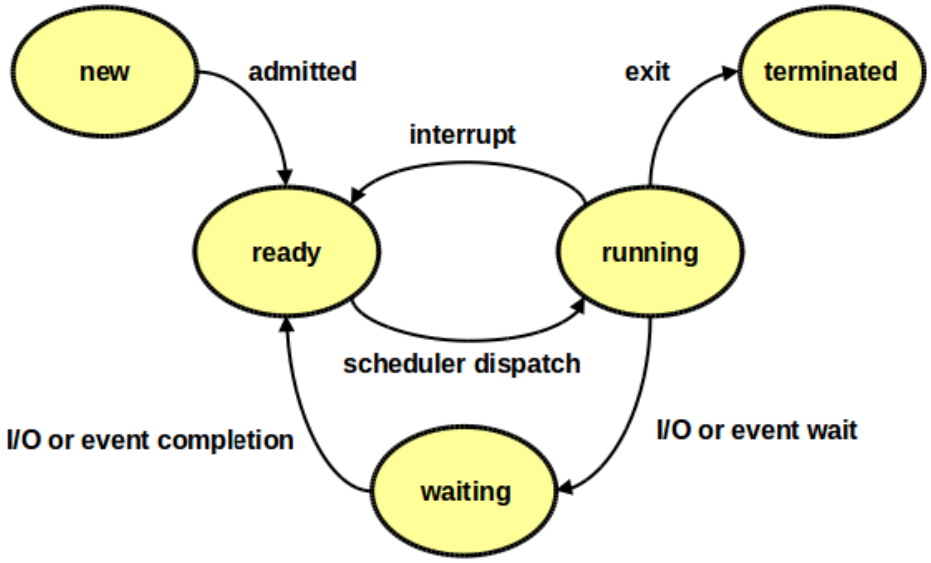
7. Plánování procesů

Podstata a cíle plánování úloh v operačních systémech. Realizace plánování činnosti procesorů. Uváznutí, podmínky uváznutí a metody ochrany proti uváznutí.
PB152/PB153

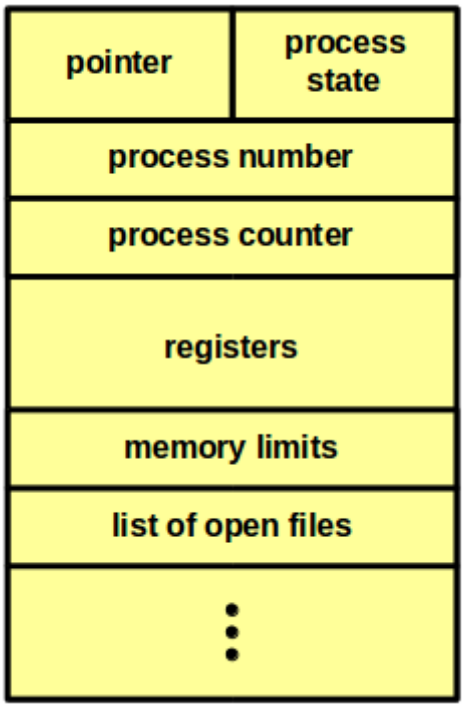
1. CO je proces?

Společné pojmenování pro spuštěný program.

Proces obsahuje: čítač instrukcí, zásobník, datovou sekci, program (instrukční sekce). Maju hierarchiu (rodic, potomok, sourozenec).



Stavy procesu: novy, beziaci, cakajuci, pripraveny, ukonceny (odlozeny cekajici, odlozeny pripravey



Process Control Block: tabulka obsahující informace potřebné pro definici a správu procesu

2. Preco?
OS musí rozhodnout, který proces (vlákno) poběží(rozhodnutí typicky optimalizuje nějakou metriku: [response time, throughput, utilization, fairness]).

Podstata

- Ukol/Funkce jádra operačního systému, ve kterém je spuštěno více procesů najednou. (Týká se tedy víceúlohových systémů podporujících multitasking, které využívají pseudoparalelismus)
- Planovac zabezpecuje prepinanie kontextu (procesov)
- Plánovač si udržuje seznamy úloh vo frontách, ty mezi nimi migrují.
    - **Fronta všech úloh** – seznam všech procesů v systému.
    - **Fronta připravených procesů** – seznam procesů uložených v hlavní paměti a připravených k běhu.
    - **Fronta odložených procesů** – seznam procesů čekajících na přidělení místa v hlavní paměti.
    - **Fronta na přidělení zařízení** – seznam procesů čekajících na I/O operace.
    - **Fronta na semafor –** seznam procesů, které čekají na synchronizační událost.

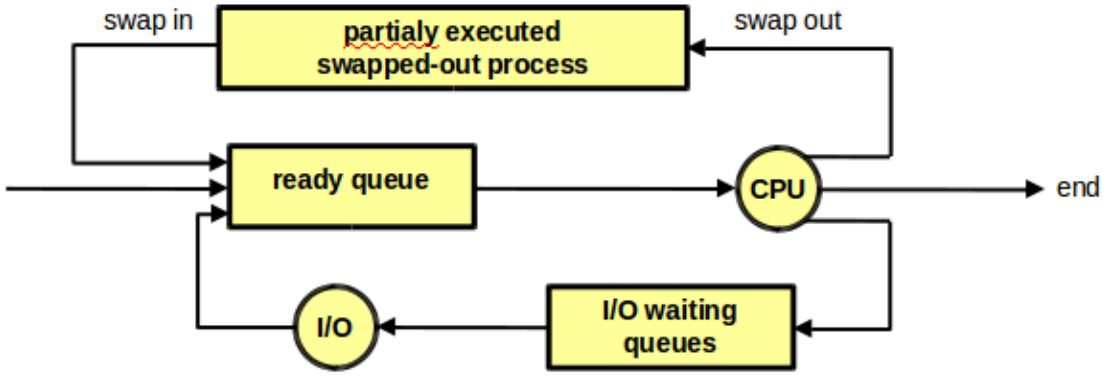
Cíle

- řeší výběr, kterému následujícímu procesu bude přidělen procesor a proces tak poběží
- výběr je závislý na prioritách jednotlivých procesů a algoritmu

Realizace plánování (aka. druhy)

- **Dlouhodobý plánovač** (strategický plánovač, job scheduler)
    - Rozhoduje, která další úloha bude spuštěna.
    - Jeho úkolem je naplánovat spouštění úloh tak, aby byl počítač maximálně využit.
    - Je vyvoláván poměrně zřídka a nemusí být rychlý – typicky při ukončení jednoho procesu rozhodne o tom, kterou úlohu dále vybrat k zavedení do paměti a spuštění.

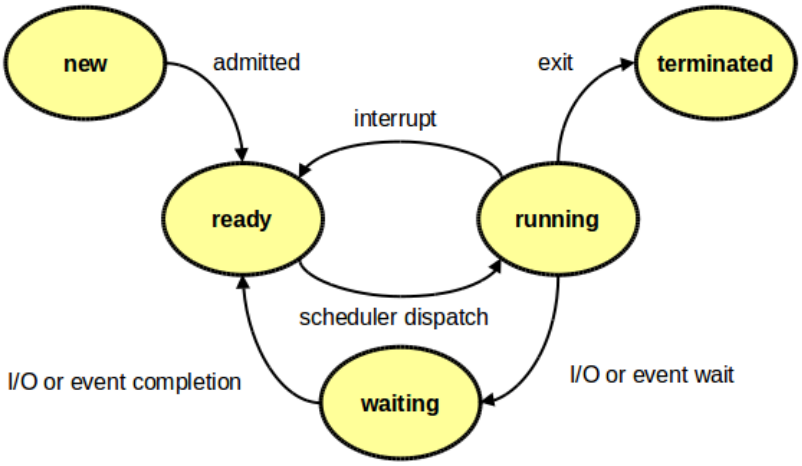
- **Střednědobý plánovač**
  - Vybírá, který připravený nebo blokovaný proces bude odsunut z vnitřní paměti na pevný disk, je-li vnitřní paměti plná (swapovaní).
  - Určuje, který proces lze zařadit mezi odložené a který odložený lze zařadit mezi připravené procesy.



- **Krátkodobý plánovač** (operační plánovač, dispečer, dispatcher)
  - Plánování procesoru – vybírá proces, který poběží na právě uvolněném procesoru.
  - Volán velice často (desítky, stovky milisekund) -> musí proto rychlý.
    - Proces přechází ze stavu běžící do stavu čekající.
    - Proces skončí.
    - Proces přechází ze stavu běžící do stavu připravený.
    - Proces přechází ze stavu čekající do stavu připravený.

      - případy *a)* a *b)* se označují jako nepreemtivní plánování (plánování bez předbíhání), pripady *c)* a *d)* se označují jako preemptivní plánování (plánování s předbíháním)

Plánování procesoru



- Plánovač v operačním systému rozhoduje, kterému procesu bude přidělen procesor
- Ako sa proces ukonci? Proces provede poslední příkaz a sám požádá OS o ukončení
- Plánování probíhá v případě, když nastane událost, která způsobí čekání procesu, nebo představuje příležitost na vykonání preempce (přerušení právě

vykonávaného procesu, nahrazení jinou úlohou a později jejího opětovného obnovení).

a)

Offline vs. online

offline znamena, ze všechny procesy jsou k dispozici od začátku a žádné již nepřibydou a zaroven o všech procesech je známo jak dlouho poběží
priklad: FIFO, SJF (shortest job first)

online znamena, ze procesy se objevují libovolně a neočekávaně a doba běhu procesů je neznámá
priklad: Round robin [preemptive, fairness, starvation-free], prioritna fronta, prioritne s viacerymi frontami, specialne (pre viac procesorov, realt-ime..)$

Algoritmy plánování

Kritéria plánování a optimalizace

- **Efektivita** – maximální využití CPU a ostatních zdrojů – cílem je udržení CPU v kontinuální činnosti.
- **Propustnost** – počet procesů, které dokončí svůj běh za jednotku času.
- **Doba obrátky** – doba potřebná pro provedení konkrétního procesu.
- **Doba čekání** – doba, po kterou proces čekal ve frontě připravených procesů.
- **Doba odezvy** – doba, která uplyne od okamžiku zadání požadavku do doby první reakce.
- **Priorita procesu**
- **Spravedlivost** - každý proces dostane spravedlivý díl času procesoru.

Algoritmus FCFS (First Come, First Served)

- Procesy jsou řazeny ve FIFO frontě – jakmile se uvolní procesor, první proces ve frontě je přiřazen procesoru a předchozí proces se zařadí na konec fronty.

+ Jednoduchý na implementaci a vhodný pro **dlouhé** procesy.
- Dlouhá průměrná doba čekání – krátké procesy jsou ovlivněny „konvojovým efektem“ (v jejich provedení brání dlouhé procesy).

Shortest-Job-First

- Vybírá se proces s nejkratším požadavkem na CPU – je nutné znát délku příštího požadavku pro každý proces (zřídka známé dopředu, používá se exponenciální průměrování na základě historie velikostí dávek procesu).
- Dvě varianty:
    - **Nepreemptivní** – jakmile proces získá přístup k CPU, tak nemůže být předběhnut.
    - **Preemptivní** (**Shortest-Remaining-Time-First**) – jakmile se ve frontě připravených procesů objeví proces s délkou dávku CPU, která je kratší než je

doba zbývající k dokončení dávky právě běžícího procesu, tak je stávající proces nahrazen novým.

Round Robin

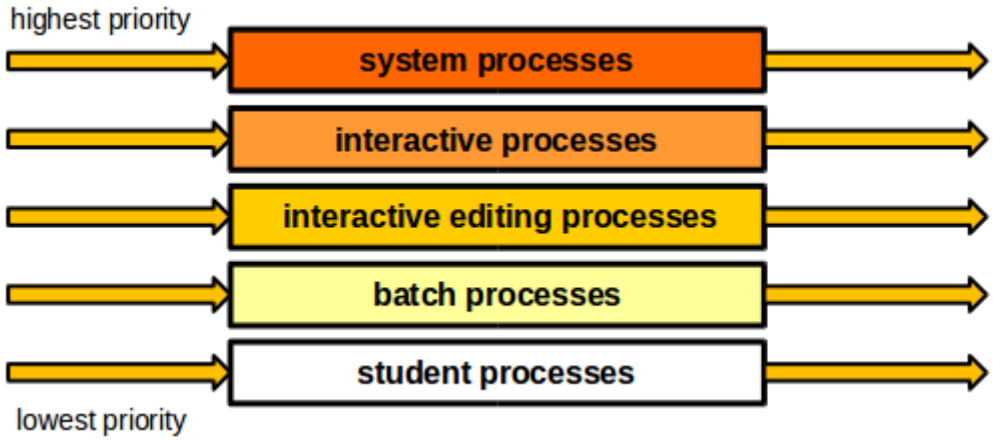
- Preemtivní algoritmus.
- Každý proces dostává CPU na malou jednotku času (desítky až stovky ms) a po jejím uplynutí je běžící proces nahrazen nejstarším procesem ve frontě a sám se zařadí na její konec.
- Efektivita silně závisí na velikosti časové jednotky přidělované procesům – příliš časté přepínání vede k velké režii operačního systému (Windows 20ms).

Prioritní plánování

- Každému procesu je přiděleno prioritní číslo (obvykle nejvyšší prioritě odpovídá nejnižší číslo) a CPU je přidělován procesu s největší prioritou.
- Dvě varianty:
  - **Nepreemptivní** – jakmile proces získá přístup k CPU, tak nemůže být předběhnut.
  - **Preemptivní** – proces může být předběhnut procesem s vyšší prioritou

- Problém stárnutí – procesy s nižší prioritou se nemusí nikdy provést, řešením je „zrání“, kdy se priorita s postupem času zvyšuje.

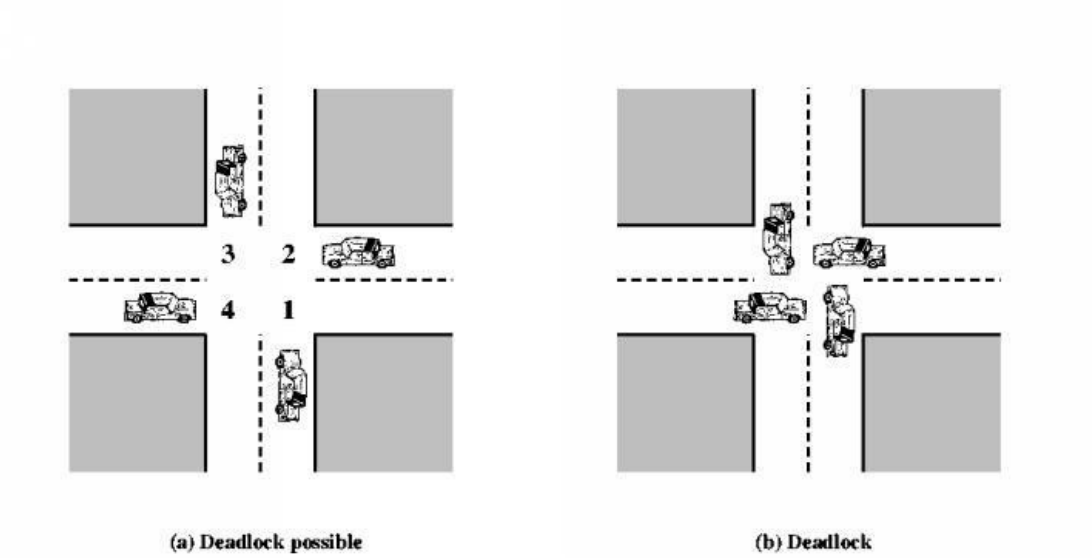
Viac front



Uváznutí a metody proti uváznutí

Preco? Synchronizacia procesov pri paralelnom behu, ci uz komunikacia alebo ziedlanie prostriedkov (vzajomne vylucenie)

- **Problém uváznutí (deadlock)** – existuje množina blokovaných procesů, kde každý proces vlastní nějaký prostředek a čeká na zdroj držený jiným procesem z této množiny.



- Livelock : ako deadlock, ale stav procesov sa stale meni (vyzera, ze nieco robia) (lock 12, a lock21)

Podmínky uváznutí

- K uváznutí dojde při splnění všech následujících podmínek (*Coffmanovy podmínky)*:
  - **Vzájemné vyloučení (mutual exclusion)** – sdílený zdroj může v jednom okamžiku používat pouze jeden proces (jinak dojde k chybě).
  - **Ponechání si zdroje a čekání na další (hold and wait)** – proces vlastnící jeden zdroj může čekat na zdroj, který drží jiný proces.
  - **Bez předbíhání (no preemption)** – jakmile proces získá zdroj, tak mu nemůže být odebrán – musí se ho vzdát.
  - **Kruhové čekání (circular wait)** – existuje seznam čekajících procesů (P0, P1, …, Pn) takový, že P0 čeká na uvolnění zdroje vlastněného P1, P1 čeká na uvolnění zdroje drženého P2, …, Pn čeká na uvolnění zdroje drženého P0

Metody ochrany proti uváznutí

- **Prevence** – zajistí se, že uváznutí nikdy nenastane – zruší se platnost některé z nutných podmínek.
- **Obcházení uváznutí** – detekuje se potenciální možnost vzniku uváznutí a takový stav se nepřipustí – zruší se současně platnost všech podmínek.

    - Prostředek se nepřidělí, pokud hrozí uváznutí (může docházet ke stárnutí).
- **Detekce uváznutí** – uváznutí povolíme, ale jeho vznik detekujeme a řešíme až následně (např. ukončení jednoho z procesů).
- **Ignorování uváznutí** – uváznutí je věc aplikace a ne systému – tento způsob řešení většinou volí OS.

Metody prevence uváznutí

- [**hold and wait+circular wait**]Proces musí požádat o všechny požadované zdroje a obdržet je ještě před tím než se spustí jeho běh nebo o ně smí žádat, pouze pokud žádný zdroj nevlastní (co je stale tazke).
- [**hold and wait+circular wait**]Pokud proces držící nějaké zdroje požaduje přidělení dalšího zdroje, který mu nelze přidělit okamžitě (není aktuálně volný), pak jsou mu odebrány všechny jím držené zdroje a proces je obnoven ve chvíli, kdy může získat všechny požadované zdroje najednou.
- [**mutual exclusion**] lock-free pristup: TestAndSet[ABa problem], atomin Swap

Detekce a obnova uváznutí

- Využití grafu čekání (wait-for graph) – jednotlivé procesy představují uzly grafu a hrana z uzlu A do B existuje, pokud proces A čeká na proces B, periodicky se poté provádí algoritmus, který v grafu hledá cykly (složitost O(|V| + |E|), kde |V| je počet uzlů a |E| je počet hran)
- Metody obnovy:
    - **Ukončení procesu** – násilně se ukončují uváznuté procesy, dokud se neodstraní cyklus (pořadí je dáno např. prioritou procesu nebo prostředky, které proces použil).
    - **Nové rozdělení prostředků** – návrat do některého bezpečného stavu (stav, kdy je možné alokovat zdroje každému procesu v nějakém pořadí a nenastane uváznutí) a restart procesu z tohoto stavu.