OPERAČNÍ SYSTÉM A PLÁNOVÁNÍ PROCESŮ

- **Charakteristika operačního systému**
- **RTOS**
- **Typy jader operačního systému**
- **Proces vs. Vlákno**
  - **Charakteristika**
  - **PCB**
  - **TCB**
- **Přepínání kontextů**
- **Plánovače OS**
  - **Preemptivní a nepreemptivní plánování**
- **Plánovací algoritmy**
  - **FCFS**
  - **SJF**
  - **SRTF**
  - **RR**
  - **PS**
  - **MFQS**

Operační systém

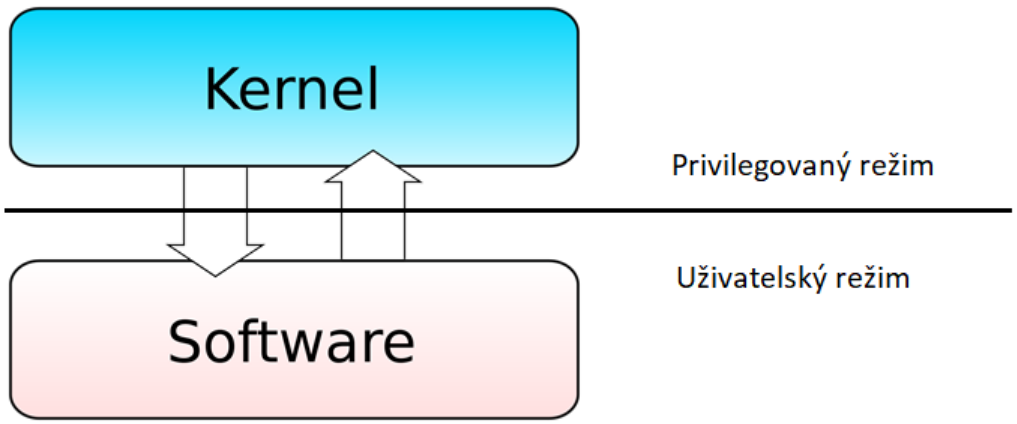
- OS je kolekce programů, která spojuje hardware s uživatelskými programy
- OS je základní vybavení PC
- OS se zavádí do operační paměti RAM při startu PC a tam zůstává až do jeho vypnutí PC
- OS zajišťuje
    - Spuštění a správu CPU
    - Správu operační paměti, systému
    - Obsluhu virtuální a vnější paměti
    - Obsluhu uživatelského rozhraní
- OS se skládá z jádra, ovladačů, příkazového procesoru, grafické nadstavby (GUI) a podpůrných programů

Jádro OS

- Říká se mu CORE nebo KERNEL
- Je to nejnižší a nejzákladnější část OS
- Je zaveden jako první při startu OS do operační paměti, zůstává tam až do vypnutí PC
- Jádro běží v privilegovaném režimu -> nikdy neztratí kontrolu nad PC
- Jádro provádí operace nad HW
- Navazuje přímo na HW a pro uživatele je zcela zapouzdřen
- Systém a uživatelské programy žádají jádro o služby prostřednictvím systémových volání
    - KERNEL INTERFACE
        - Přímé volání pomocí specializované instrukce
    - LIBOVOLNÝ INTERFACE
        - Volání funkce ze systémových knihoven

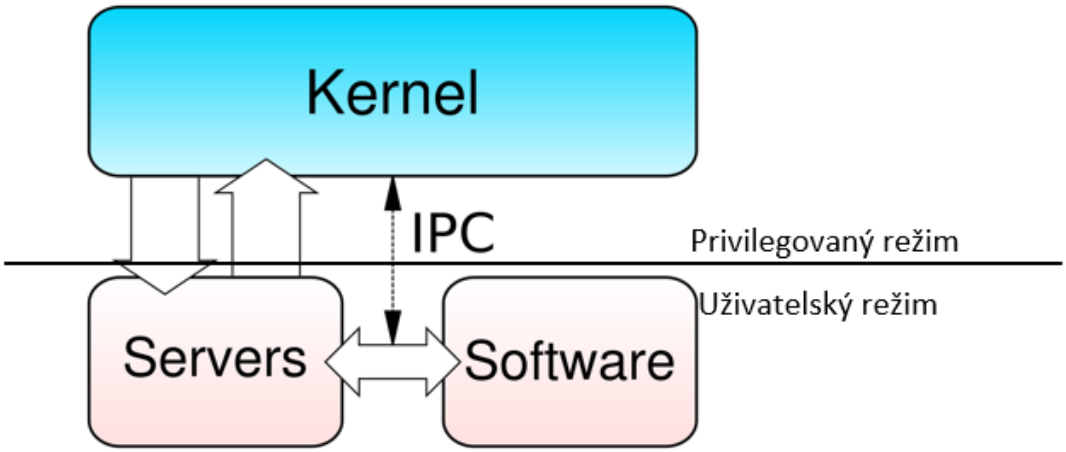
Typy jader OS

- **Monolitické jádro**
  - Veškerý kód běží v privilegovaném režimu za účelem vysoké efektivity
    - Správa paměti
    - Meziprocesová komunikace
    - Souborový systém
    - Síťová komunikace
  - Chyba v jednom subsystému může ovlivnit další
  - Vylepšením této koncepce je dynamické nahrávání modulu
    - Modul je možno přidat za běhu – bez restartu
      - Ovládání USB disku
    - Moduly jsou zavedeny do adresových prostorů jádra, kde se připojí s jeho funkcemi (privilegovaný režim)
    - Dochází ze zpoždění
  - Představitelé
    - MS-DOS, WIN 95/98, Mac OS do 8.6 (nemodulární)

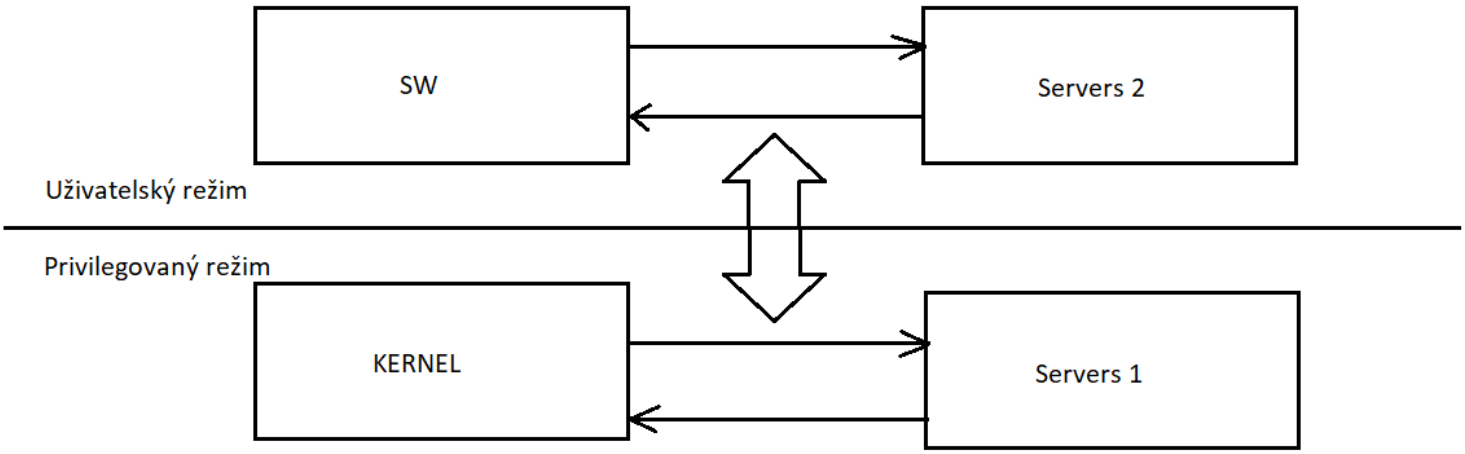


    - Free VSD, Linux (modulární)

- **Micro jádro**
  - Minimalizuje rozsah jádra a nabízí jednoduché rozhraní s jednoduchými abstrakcemi a malým počtem služeb pro nejzákladnější správu CPU, I/O zařízení, paměti a IPC
  - Ostatní služby jako jsou plánování, správa SW systému, ovladače zařízení jsou implementovány mimo jádro v podobě serveru
  - Výhody
    - Jednodušší programování díky rozdělení na logické celky
    - Flexibilita – možnost více současně běžících implementací služeb
    - Bezpečnost - v případě napadení, nehrozí pád PC
  - Nevýhoda
    - Vysoká režie IPC = meziprocesová komunikace
  - Minix, Symbian OS



- **Hybridní jádro**
  - Kombinace monolitického a micro jádra
  - Objevuje se v dnešních OS
  - Micro jádro je rozšířeno o kód, který by mohlo běžel v podobě serveru v uživatelském režimu, ale za účelem zmenšení režie IPC je těsněji provázán a běží v privilegovaném režimu v podobě serveru
  - Nedokáže za běhu samo zavádět moduly
  - Windows, Mac OS X



- **Exo jádro**
  - Experimentální jádro
  - Používá se na universitách k testování
  - Poskytuje velmi nízké rozhraní zamřené na bezpečné sdělení prostředků
  - Menší než mikrojádra
  - AE GIS, NEMESIS

- **Nano jádro**
  - Menší než mikro jádro
  - Služby jsou v něm řešeny jako ovladače

RTOS

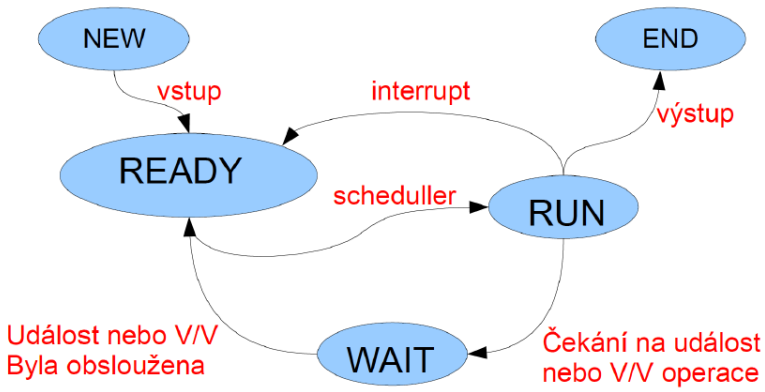
- Real Time Operating Systém
- Jednoduché vestavěné systémy jsou ovládány pomocí supersmyčky
    - Obsahuje globální proměnné
    - Nejjednodušší způsob, jak naprogramovat CPU
    - Každý krok musí čekat, než na něj příjde řada
- Složité systémy používají RTOS
- Jednotlivé programové funkce tvoří samostatné úlohy – tasky
- Provádění úloh řídí krátkodobý plánovač – scheduler
- OS, který poskytuje možnost reagovat na události v okolí počítače průběžně (tj. v reálném čase)
- RTOS poskytuje uživateli (nebo programátorovi) záruky, že je určitou činnost v určitém časovém úseku možné dokončit
- RTOS je používán například v embedded systémech, robotice, automatizaci, elektronických měřeních nebo v telekomunikacích
- Výhody RTOS – RTX Keil
    - Task Scheduling
        - Tasky jsou volány v případě potřeby zajistit lepší výkon a odpověď na událost
    - Multitasking
        - Provádění několika tasků současně
    - Deterministické chování
        - Události a přerušení jsou zpracovány ve vymezeném čase
    - Každý task je přidělen určitému místu zásobníku, umožnuje předvídatelné využití paměti

Proces

- Proces = program je název pro spuštěný (běžící) PC program
- Proces je umístěn v operační paměti v podobě sledu strojových instrukcí vykonávané procesorem
- Obsahuje strojový kód a dynamicky měnící se data, které procesor zpracovává
- Jeden program může v PC běžet jako více procesů s různými daty (více krát spuštěný web zobrazující různé stránky)
- Správu procesů vykonává OS, který zajištuje jejich běh, přiděluje jim systémové prostředky PC a umožnuje uživateli procesy spravovat – spouštět, ukončovat,…
- Proces je v OS definován
    - Identifikátorem – PID
    - Programem, který je řízen
    - Obsahem registru – čítač instrukcí, adresa zásobníku
    - Daty
- Procesů běží v OS spousty a je nutné je spravovat
    - Proces managment – správa procesů
    - Přepínání kontextů - velmi náročné (vznik vláken)
    - Plánovač = dispatcher – plánuje na základě plánovacího algoritmu
    - Správa paměti
    - Podpora meziprocesorové komunikace

- Proces se může nacházet v 5 stavech
    - NEW – vytvořený, nový
        - Proces je vytvořen buď příkazem uživatelem, nebo na žádost OS
        - Strojový kód je schedulerem zaveden do operační paměti
    - READY – připravený
        - Proces je připravený pro vstup do stavu run, čeká pouze na přidělení procesoru
    - RUN – běžící
        - Procesu je přidělen procesor a právě provádí příslušné operace
    - WAIT – čekající, blokovaný
        - Proces je převeden do tohoto stavu v případě, kdy čeká na dokončení nějaké vstupně-výstupní operace, případně na skončení jiného procesu
        - fronta procesů čekajících na vstupní, nebo výstupní události anebo, na nějaký signál, po obsloužení se přesunou do fronty ready
    - END – ukončený
        - Proces se zpracoval a je ukončený

Diagram stavů procesu

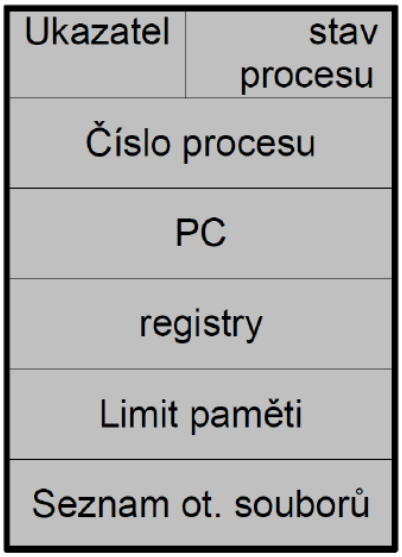


Životní cyklus procesů

- Životní cyklus procesu probíhá podle diagramu stavových přechodů
- U více procesů je zařazení k běhu řízeno pravidly
    - Časové kvanta, priorita, bez možnosti přerušení
- Proces je vytvořen příkazem uživatele, nebo na žádost OS, tento proces se registruje do fronty připravených procesů
- Takto vytvořené procesy jsou ve stavu READY, čekají na přidělení procesoru
- Na základně plánovacího algoritmu vstoupí proces do RUN a začne se vykonávat
- Běžící program může být ukončen normálně, tzn., provede se celý, nebo násilně ukončen uživatelem, provedením chybné strojové instrukce, chybou vstupně-výstupního zařízení, porušením ochrany paměti, nebo na žádost rodiče, ….
- Běžící proces může po uplynutí časového limitu pro jeho běh převeden do stavu READY
- Běžící proces může být jen jeden, máme-li jeden procesor, ve stavu READY může být více procesů

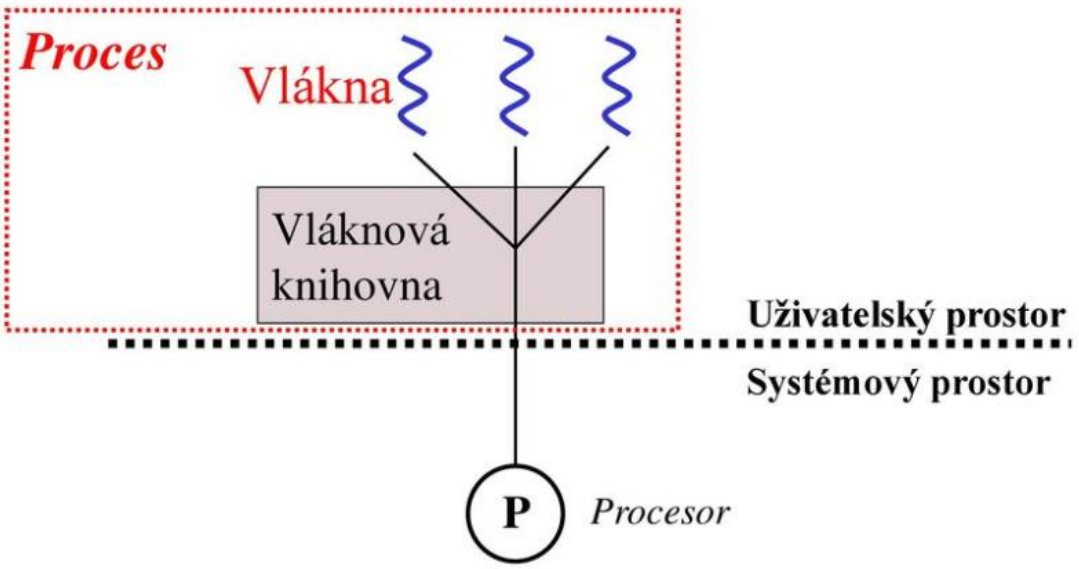
Řídící blok = Proces Control Block

- PCB je datová struktura v jádře OS, která obsahuje informace potřebné pro správu a běh procesu
- Každý proces má svůj vlastní PCB, přičemž jejich maximální počet může být dán jádrem OS nebo mohou být vytvářeny dynamicky
- PCB je umístěn v části paměti, která je chráněna před přístupem ostatních uživatelů a procesů z důvodu obsahu důležitých informací
- V některých OS je PCB umístěn na začátku zásobníku jádra OS pro daný proces, protože je to vhodně chráněné místo
- Obsah PCB
    - Ukazatel
        - Odkaz na další PCB
    - Stav procesu
        - V jakém stavu momentálně proces je
    - Číslo procesu – PID
        - Jedinečné číslo procesu, pod kterým vystupuje
    - PC – Program Counter
        - Adresa následující strojové instrukce
    - Registry
        - Registry procesoru
    - Limit paměti
        - Adresní prostor procesu
    - Seznam otevřených souborů
        - I/O info
    - Priorita, ukazatel na paměť, kdy naposled běžel, jak dlouho,…

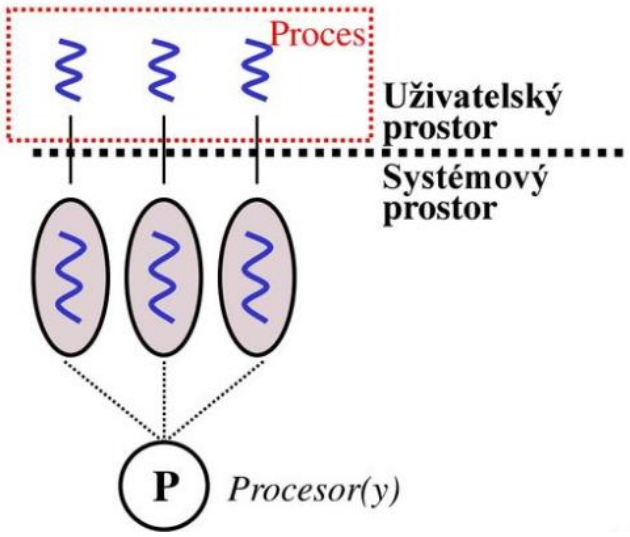


Vlákno

- Vlákno je odlehčený proces a je jeho součástí – sám nemůže existovat
- Vlákno má životní cyklus
- Ukládají se do TCB – Threat Control Block
- Pomocí vlákna se snižuje režie OS při změně
- Vlákna mezi sebou sdílejí stejnou paměť
- Vlakna mají stejná práva jako proces
- Spotřebovávají méně paměti než celý proces
- Rychlejší přepínání na uživatelské úrovni
- Použití např. při vykreslování grafiky, excel – jedno vlákno provádí instrukce, druhé zobrazuje
- TCB – Threat Control Block
    - Tabulka vlákna nacházející se v jádře OS
    - Datová struktura obsahující informace o vláknu
        - Jednoznačný identifikátor vlákna – TID
        - Ukazatel na proces - PCB, ke kterému se vztahuje (žije v něm)
        - Ukazatel na aktuální programovou instrukci vlákna
        - Stav vlákna
        - Registry
- Vlákno na uživatelské úrovni
    - Tím, že jsou vlákna spravována na uživatelské úrovni, tak o nich OS neví – jsou nezavislé
    - Není nutno volat jádro OS pro práci s vlákny
    - Jsou v plné režii programátora
    - Výhody:
        - Rychlé přepínané mezi vlákny
        - Uživatelský proces má nad vlákny plnou kontrolu
    - Nevýhody:
        - Vlákno neví o jádrech, tudíž přiděluje procesorový čas procesům a není možné, aby 2 vlákna stejného procesu běžela současně a to i v případě víceprocesorového systému



- Vlákno na úrovni jádra OS
    - O všechno se stará jádro OS
    - Jeden proces může využívat více procesorů
    - Volání služby neblokuje ostatní vlákna z CPU
    - Náročnější správa
    - Nespravedlivé plánování – CPU čas je přidělován vláknum a ne procesu



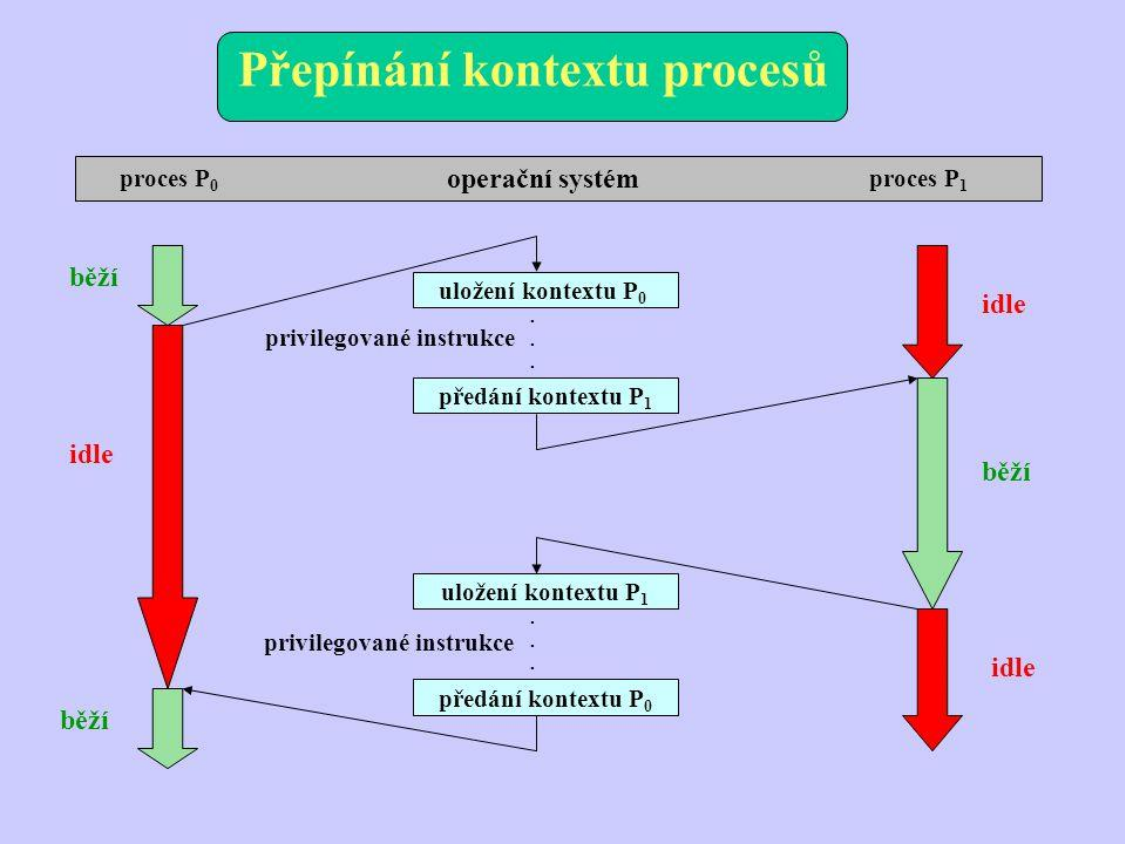
- Použití vláken
    - Obsluha periferií – 1 vlákno komunikace s HW, 2 vlákno komunikace s uživatelem
    - U serverů pro každého uživatele, který se připojí na server, bude vyhrazené 1 vlákno
- Výhody vláken:
    - Menší režie
    - Urychlení výpočtu, odezvy programu, celkového běhu
    - Efektivní využití systému
    - Jednodušší sdílení a komunikace než mezi procesy
    - Paralelní běh
    - Lepší a přehlednější strukturalizace programu
- Nevýhody vláken
    - Omezeni počtu vytvořených vláken
    - Náročnější kód pro řešení souběhu vláken a pro sdílení prostředků

PCB vs TCB

- Oba se nacházejí v jádře OS a obsahují stejné informace
- Jediný rozdíl je v tom, že TCB má navíc pár specifických hodnot jako například ukazatel na proces, který obsahuje dané vlákno

Přepínaní kontextů

- Jedná se o operaci, která přepne jeden běžící kontext na druhý
- Opakuje se několikrát za sekundu (v rádu ns)
- Obsahují všechny multitaskingové OS
- Dochází k němu i při obsluze přerušení, nebo při změně režimu (priv -> uživ)
- O přepnutí kontextů se stará dispatcher na základně nastavené plánovače – scheduleru
- Přepnutí musí být velmi rychlé, aby nebylo poznat „zamrznuzti“ systému
- Užitaveli se jeví, že všechny procesy běží najednou
- Přepnutí může být SW i HW
    - SW je přesnější, ale pomalejší, HW je rychlejší ale méně přesné

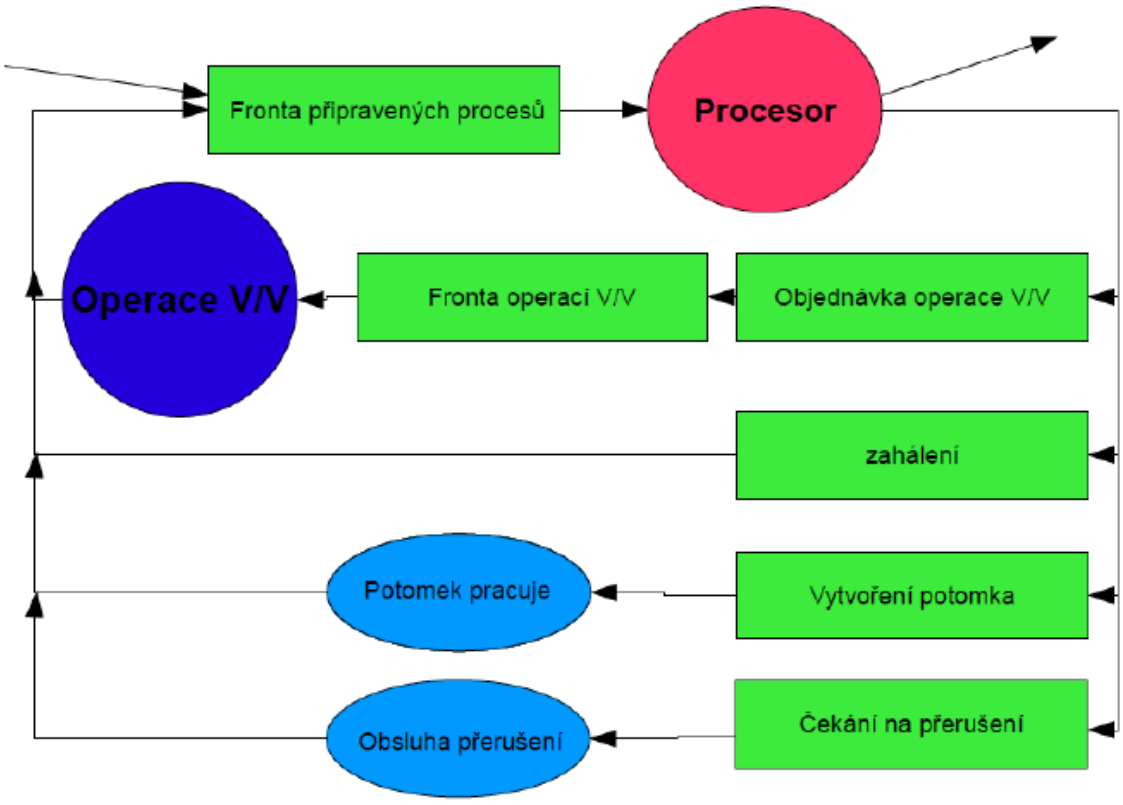


- Fronta připravených procesů je ve skutečnosti fronta ukazatelů na PCB prvního a posledního procesu
- Scheduler vstoupí do fronty přes začátek a vybere vhodný proces, který vyřadí z fronty a následně je spuštěn (dojde k přepnutí kontextu)
- Front je v OS několik
    - Operace pro práci z diskem
    - Vstup z klávesnice
    - Terminál

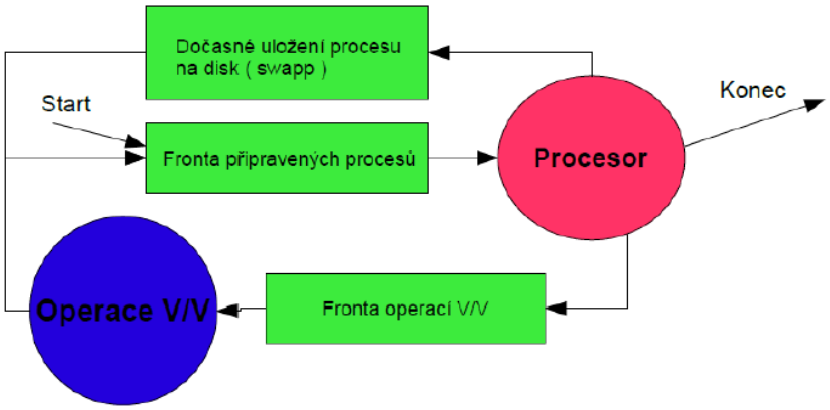
Plánovače (scheduler)

- Má za úkol určit, které úloze (procesu) bude procesor přidělen
- Krátkodobý, dlouhodobý, střednědobý

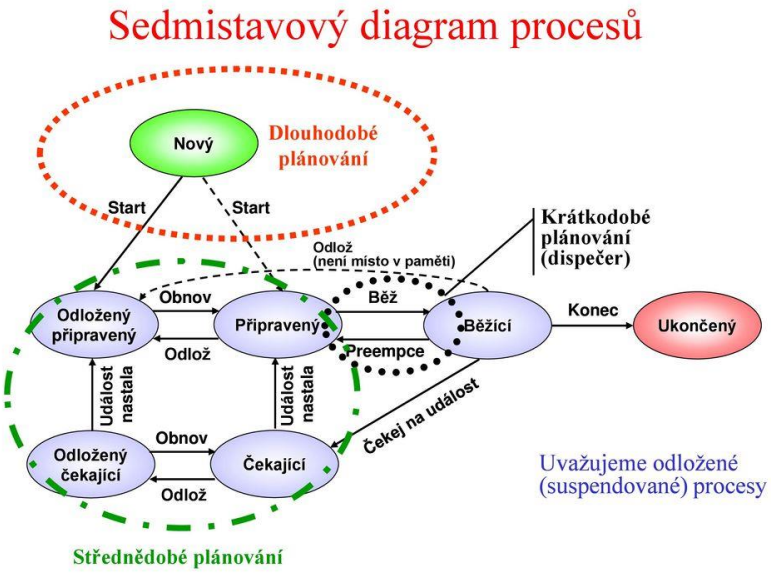
- **SHORT TURN – KRÁTKODOBÝ**
    - Velmi rychlý plánovač označován jako plánovač CPU
    - Využívá dvou front
        - Fronta připravených procesů – READY
        - Fronta I/O operací
    - Zpracování probíhá paralelně
    - U běžícího programu muže nastat:
        - Požaduje I/O opraci
        - Proces vytvoří potomka a čeká na jeho dokončení
        - Procesu je násilně odebrán CPU z důvodu přerušení
        - Čeká na událost (výstup z jiného procesu)
            - Pokud je prázdná fronta, CPU zahálí – pracuje naprázdno
            - Vytvoření potomka probíhá systémovým voláním FORK



- **Střednědobý plánovač – MID TERM SCHEDULER**
  - Taktický plánovač
  - V případě, když si začne nějaký běžící procesů žádat o větší paměťový prostor, zareaguje taktický plánovač
  - Vybírá ten proces, který je možné dačasně odložit na disk (swapování) a volá místo v paměti pro náročnější proces, zároveň vybírá z uložených procesů na disku ten, který vrátí zpět do OP -> fronta READY
  - Vznik dvou nových stavů
    - Odložený čekající = swap wait
    - Odložený připravený = swap ready



- **LONG TERM – Dlouhodobý plánovač**
  - Strategický plánovač
  - Není tak rychlý, aktivace řádově v jednotkách desítek sekund
  - Jde o vhodnou kombinaci několika úloh náročných na I/O úloh a CPU
  - V interaktivních systémech (Win, Mac, Linux) se prakticky nepoužívá, zastupuje je krátkodobý plánovač
  - Využit v RTOS systémech



Plánování procesoru

- Scheduler se rozhoduje, kterému procesu přidělí procesor na základě
    - Ukončení procesu – nepreemptivní
    - Změně stavu procesu z RUN do READY – preemptivní
    - Změně stavu procesu z RUN do WAIT – nepreemptivní
    - Změně stavu procesu z WAIT do READY – preemptivní
- UNIX používá preemptivní plánování, v některých situacích zakazuje přerušení
- WINDOWS používá nepreemptivní plánování, nepotřebuje speciální HW (časovač)
- Preemptivní a nepreemptivní plánování
    - **PREEMTIVNÍ PLÁNOVÁNÍ**
        - OS má plnou kontrolu a může kdykoliv odebrat procesu procesor
        - Kontrola nap PC a nad všemi prostředky
        - Plánování s přebíháním
        - Dochází k němu při uplynutí přidělené doby využití CPU -> vyvolano přerušeném časovače
        - Nedochází k zamrznutí PC
        - Nevýhoda je složitější implementace a nutnost HW podpory CPU
        - Mac OS X, Linux
    - **NEPREEMPTIVNÍ PLÁNOVÁNÍ**
        - OS nemá plnou kontrolu nad převzetím CPU -> musí počkat, až mu procesor sám nabídne převzetí
        - Nelze násilně odebrat CPU procesu, ani jiné systémové prostředky
        - Plánování bez předbíhání
        - Výhoda je jednodušší implementace
        - Použití v uzavřených systémech -> všechny procesy jsou předem známy i jejich vlastnosti a jsou naprogramovány tak, aby samy uvolňovaly procesor pro další procesy
        - Windows
- Cíle plánování:
    - Využití CPU – maximalizace kontinuální činnosti CPU
    - Propustnost – maximalizace ukončených procesů za jednotku času
    - Doba čekání – minimalizace dobý čekání procesů ve frontě ready
    - Do obrátky – minimalizace dob potřebné k provedení procesu
    - Doba odpovědi – minimalizace doby, která uběhne od okamžiku zadání požadavku do jeho první reakce, nejedná se o dobu do úplného výpisu, čili následku běhu celého programu
    - Podle váhy kritérií se rozlišují plánovací algoritmy, tzv. strategie plánování procesoru

Plánovací algoritmy

- **FCFS = FIRST CAME FIRST SERVE**
  - Proces, který přišel jako první, bude první obsloužen
  - Nepreemptivní plánování
  - Dlouhé procesy blokují krátké -> vznik kolon
  - Samostatně se téměř nepoužívá
  - Velká průměrná čekací doba

PROCES	AT	BT	CT	TAT	WT	VT
P0	***0***	***4***	*4*	*4*	*0*	*0*
P1	***2***	***7***	*11*	*9*	*2*	*4*
P2	***5***	***2***	*13*	*8*	*6*	*11*
P3	***6***	***1***	*14*	*8*	*7*	*13*
P4	***8***	***3***	*17*	*9*	*6*	*14*

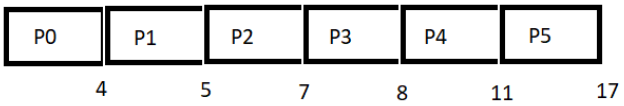
AT (Arriving Time) – příchod procesu
BT (Burst Time) – délka trvaní procesu
CT (Complition Time) – doba ukončení
TAT (Turn Around Time) – čas obrátky
WT (Waiting Time) – doba čekání
VT (Visiting Time) – kdy poprvé se objevil proces na procesoru
TAT = CT – AT
WT = TAT – BT
VT = dá se zjistit

- **SJF = SHOTEST JOB FIRST**
  - Proces, který má nejkratší požadavek na CPU bude zpracován jako první
  - Nepreemptivní plánování – proces nemůže být přerušen, musí se počkat, až se dokončí
  - V případě, že se objeví dva procesy se stejnou dobou zpracování (BT), rozhodne FCFS
  - U této strategie musíme dopředu znát délku příštího požadavku pro každý proces
  - Nevznikají kolony
  - Hrozí hladovění s procesy s dlouhým WT

PROCES	AT	BT	CT	TAT	WT	VT
P0	***0***	***4***	*4*	*4*	*0*	*0*
P1	***2***	***7***	*11*	*9*	*2*	*4*
P2	***5***	***2***	*14*	*9*	*7*	*12*
P3	***6***	***1***	*12*	*6*	*5*	*11*
P4	***8***	***3***	*17*	*9*	*6*	*14*

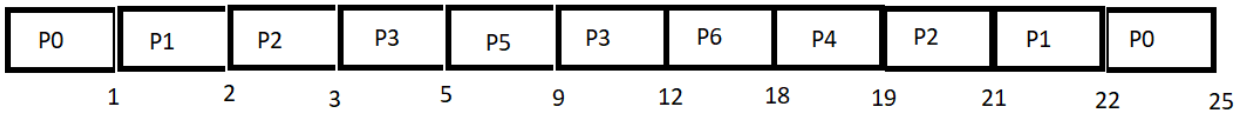
- **SRTF = Shortest Remaining Time First**
  - Proces, který má nejkratší požadavek na CPU se zpracuje první
  - Preemptivní verze SJF
  - Proces se zpracovává, dokud není ukončen nebo se neobjeví nový proces s kratší dobou zpracování
  - Nevznikají kolony – krátké procesy se zpracovávají nejdřív
  - Vyhladovění dlouhých procesů (řešení je zvýšit prioritu procesu a zvýšit tak šanci na jejich proběhnutí)

proces	AT	BT	CT	TAT	WT	VT
P0	***0***	***4***	*4*	*4*	*0*	*0*
P1	***2***	***7***	*17*	*15*	*8*	*4*
P2	***5***	***2***	*7*	*2*	*0*	*5*
P3	***6***	***1***	*8*	*2*	*1*	*7*
P4	***8***	***3***	*11*	*3*	*0*	*8*



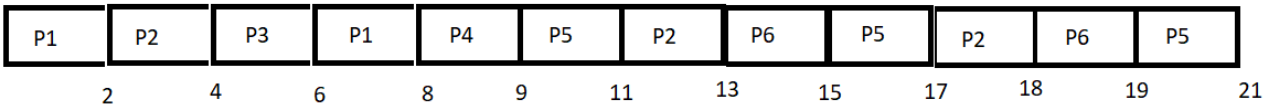
- **PS = PRIORITY SCHEDULING**
  - Preemptivní plánování s prioritou
  - Proces s nejvyšší prioritou bude zpracován jako první (každá proces má vlastní prioritu)
  - Nižší číslo = vyšší priorita
  - Pokud mají dva procesy stejnou prioritu, dřív bude zpracován ten, který přišel dřív
  - Hrozí hladovění procesů s nízkou prioritou (řešením je zvyšování priority procesům, které dlouho čekají)

Proces	P	AT	BT	CT	TAT	WT	VT	RT
P0	***2***	***0***	***4***	*25*	*25*	*21*	*0*	*0*
P1	***4***	***1***	***2***	*22*	*21*	*19*	*1*	*0*
P2	***6***	***2***	***3***	*21*	*19*	*16*	*2*	*0*
P3	***10***	***3***	***5***	*12*	*9*	*4*	*3*	*0*
P4	***8***	***4***	***1***	*19*	*15*	*14*	*18*	*14*
P5	***12***	***5***	***4***	*9*	*4*	*0*	*5*	*0*
P6	***9***	***6***	***6***	*18*	*12*	*6*	*12*	*6*



- **RR = ROUND ROBIN**
  - Jeden z nejzákladnějších implementovaných plánovacích algoritmů
  - Přikazuje běžícímu procesu časové kvantum
  - Po uběhnutí tohoto času se proces odstaví a místo něj je spuštěn další v pořadí
  - Odstavený proces jde na konec do fronty READY
  - Pokud se proces vykoná dřív, než je časové kvantum, jde ihned po jeho dokončení na řadu další
  - Nehrozí hladovění
  - Priorita je u všech procesů stejná
  - Preemptivní plánování

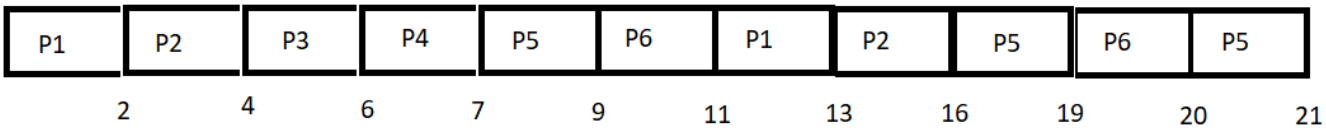
Proces	AT	BT	CT	TAT	WT	VT	RT
P1	***0***	***4***	*8*	*8*	*4*	*0*	*0*
P2	***1***	***5***	*18*	*17*	*12*	*0*	*1*
P3	***2***	***2***	*6*	*4*	*2*	*4*	*2*
P4	***3***	***1***	*9*	*6*	*5*	*8*	*5*
P5	***4***	***6***	*21*	*17*	*11*	*9*	*5*
P6	***6***	***3***	*19*	*13*	*10*	*13*	*7*



Fronta: P1, P2, P3, P1, P4, P5, P2, P6, P5, P2, P6, P5

- **MFQS = MULTILEVEL FEEDBACK QUEUE SCHEDULER**
  - Plánovač se zpětnou vazbou
  - Pracuje tak, že zařazuje procesy do řad a každá řada má jinou prioritu a časové kvantum
  - Jednotlivé řady jsou Round Robin s rozdílným časem zpracování
  - Poslední řada je FCFS
  - Procesy se zpracovávají v řadě
  - Po uplynutí stanového časového kvanta se proces posouvá no nižší řady
  - Proces s nejvyšší prioritou (nejnižší číslo) a první v řadě jde první
  - Proces, u kterého začne docházet k hladovění, může přeskočit do vyšší řady a přeskočit ostatní procesy

Proces	AT	BT	BTR	CT	TAT	WT	VT	RT
P1	***0***	***4***	*2, 0*	*13*	*13*	*9*	*0*	*0*
P2	***1***	***5***	*3, 0*	*16*	*15*	*10*	*2*	*1*
P3	***2***	***2***	*0*	*6*	*4*	*2*	*4*	*3*
P4	***3***	***1***	*0*	*7*	*4*	*3*	*6*	*3*
P5	***4***	***6***	*4, 1, 0*	*21*	*17*	*11*	*7*	*3*
P6	***6***	***3***	*1, 0*	*20*	*14*	*11*	*9*	*3*



Fronta 1 *TQ = 2 ns*

- P1, P2, P3, P4, P5, P6

Fronta 2 *TQ = 3 ns*

- P1, P2, P5, P6

Fronta 3 *FCFS*

- P5