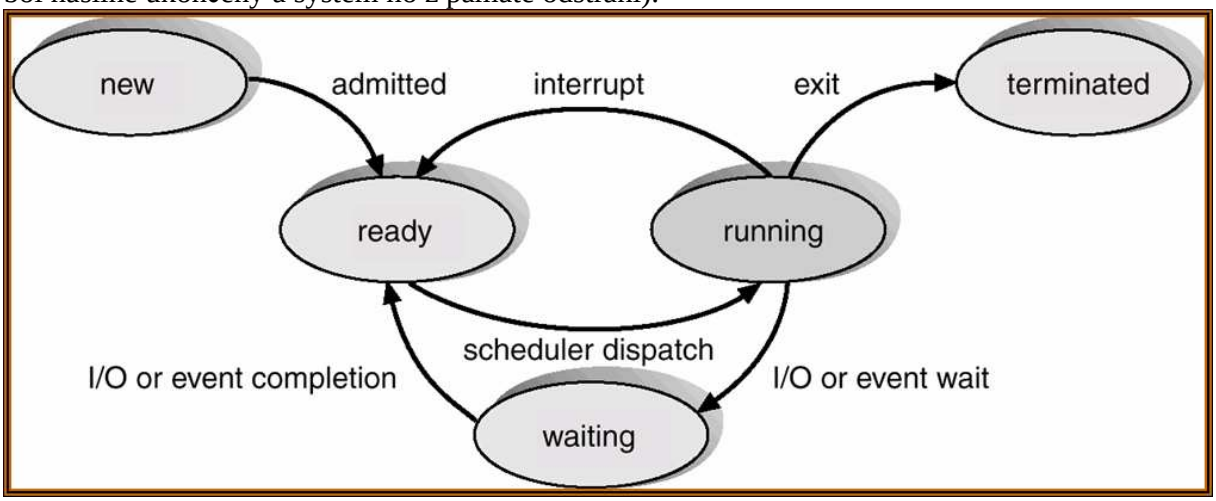
Správa procesov

Zavádzanie a vykonávanie úloh

Proces je úloha (program) zavedená do pamäte na vykonanie. Ak desať používateľov používa napríklad textový editor, spustili **jediný** program (v pamäti sa nachádza iba raz), ale ide o desať samostatných procesov.
Proces pozostáva z kódu programu (v našom príklade jediný v pamäti) a dát procesu. Toto spolu s hodnotami registrov počítača charakterizuje proces v operačnom systéme na danom počítači.

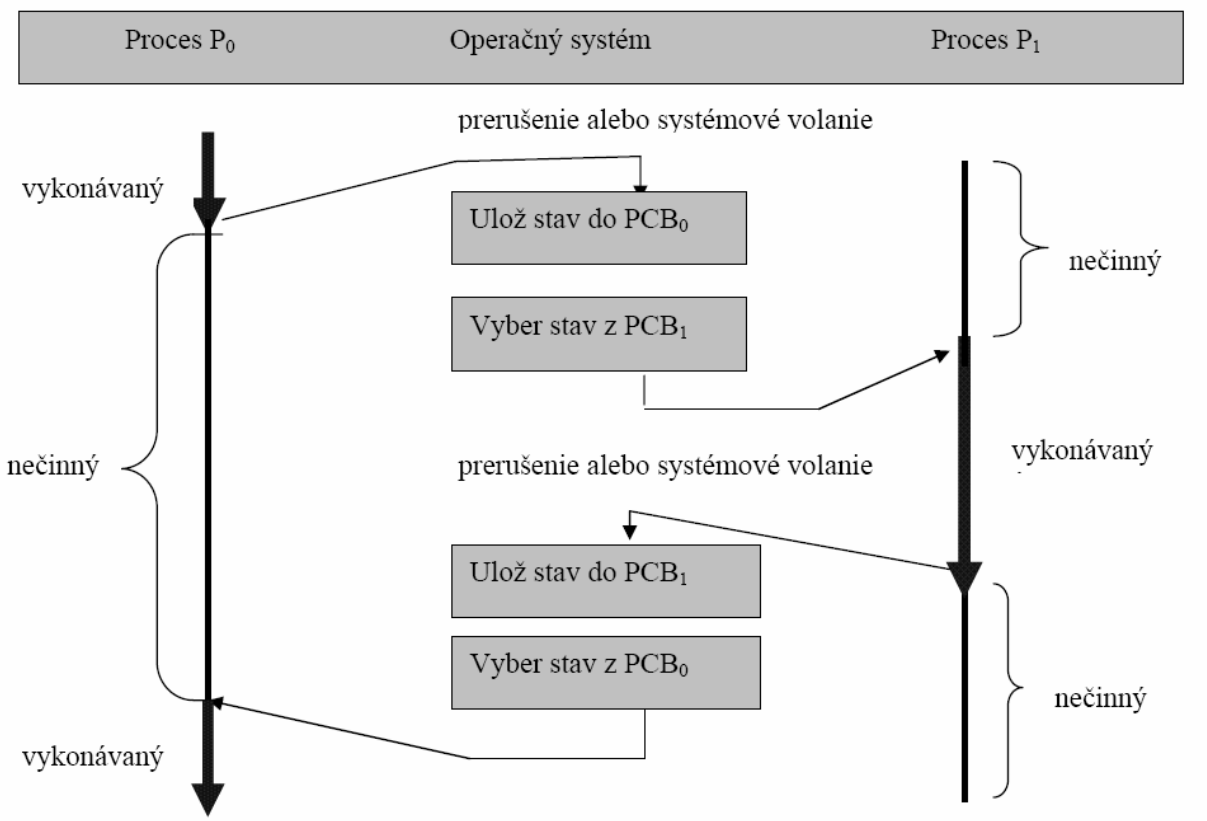
Stav procesu

Stav procesu vyjadruje vzťah OS a procesu. Proces môže byť **novovytvorený** (new), **vykonávaný** (tiež mu hovoríme bežiaci, running, práve je vykonávaný pomocou CPU), **pripravený** (ready, čaká na pridelenie CPU), **čakajúci** (waiting, nevykonáva sa, pretože čaká na dokončenie I/O požiadavky, na komuniká-ciu s iným procesom, na uplynutie zadaného časového úseku alebo na ukončenie činnosti svojho potom-ka) a **ukončený** (terminated, skončil prácu alebo bol násilne ukončený a systém ho z pamäte odstráni).



Proces má svoj životný cyklus pozostávajúci z **pripustenia** (admit) novovytvoreného procesu do stavu pripravený, **naplánovania** (dispatch) procesu – stane sa bežiacim (a je opakovane prepínaný do stavu pri-pravený podľa prioritných pravidiel) a **čakania** na výskyt udalosti (stlačenie klávesy u editora). Po výskyte udalosti, na ktorú proces čakal, sa zasa prepína do stavu pripravený. Poslednou fázou je samozrejme **ukončenie** procesu. **Za bežných okolností používateľa ani programátora životný cyklus procesu nezaujíma**.

Operačný systém si drží informácie o procesoch v špeciálnej **tabuľke procesov**. Tabuľka pozostáva z **PCB** – process control blocks. Každá položka tabuľky obsahuje ***kontext procesu***. Sú to všetky podstatné údaje o procese: **PID** (identifikačné číslo) procesu, stav procesu (či je bežiaci, pripravený, ...), obsah registrov (vrátane registra **IP** určujúceho pamäťovú adresu ďalšej inštrukcie), informácie pre plánovač procesov, údaje o pridelenej pamäti, účtovnícke informácie (koľko času CPU program spotreboval), stav I/O (priradené zariadenie, otvorené súbory).



Vlákna

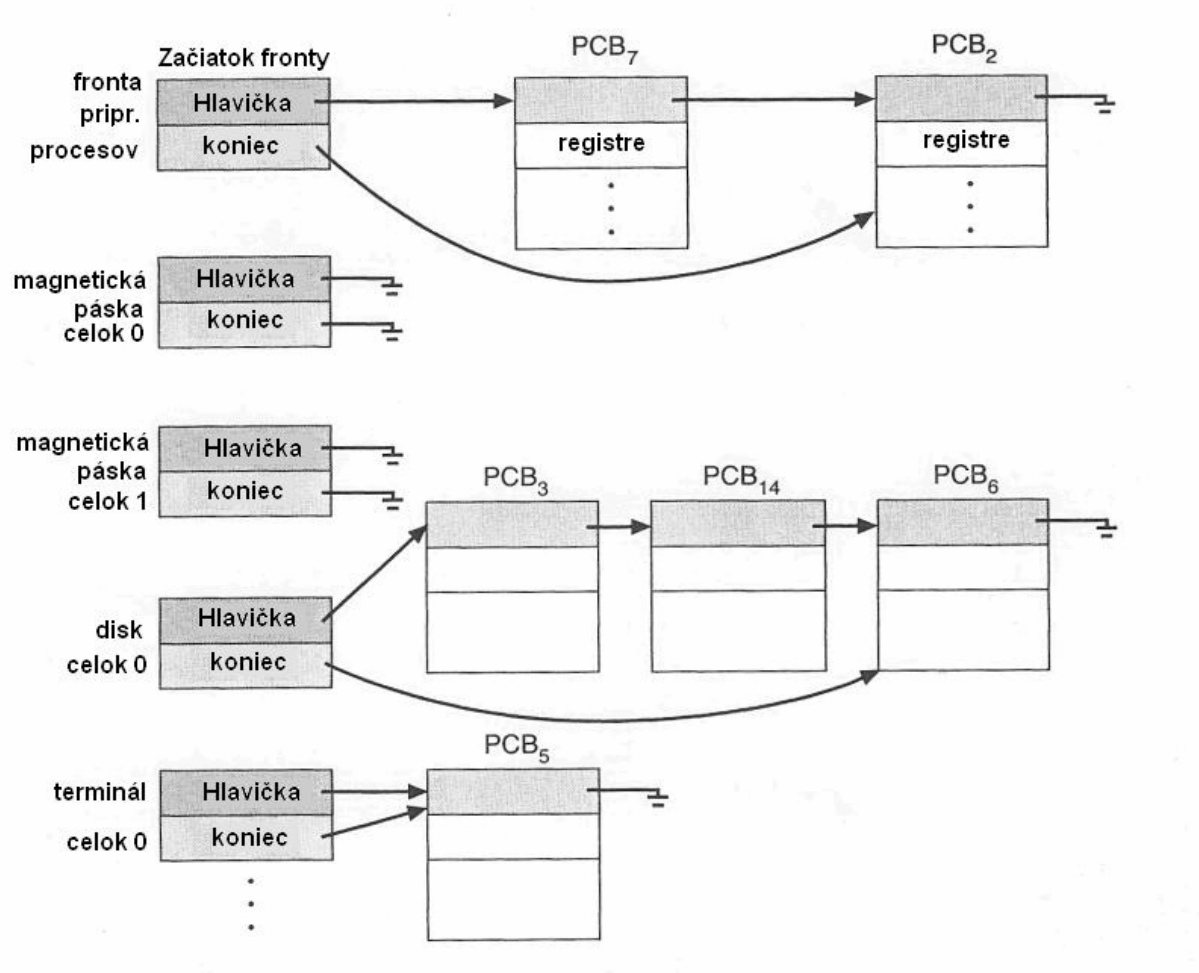
Model procesov doteraz preberaný naznačoval, že proces je program, ktorý vykonáva jedno **vlákno** (*Thread*). Napríklad: ak proces je bežanie textového editora, jedno vlákno inštrukcií je vykonávané. Toto jedno vlákno riadenia umožňuje procesu vykonávať iba jednu úlohu v jednom čase. Napríklad, užívateľ nemôže simultánne písať písmená a robiť kontrolu pravopisu v rámci toho istého procesu. Mnoho moderných operačných systémov prekonali tento koncept procesov, aby umožnili procesu obsahovať viacej vykonávaných vlákien. Týmto spôsobom umožňujú procesu vykonávať viac jako jednu úlohu v čase.

Rozvrhové fronty

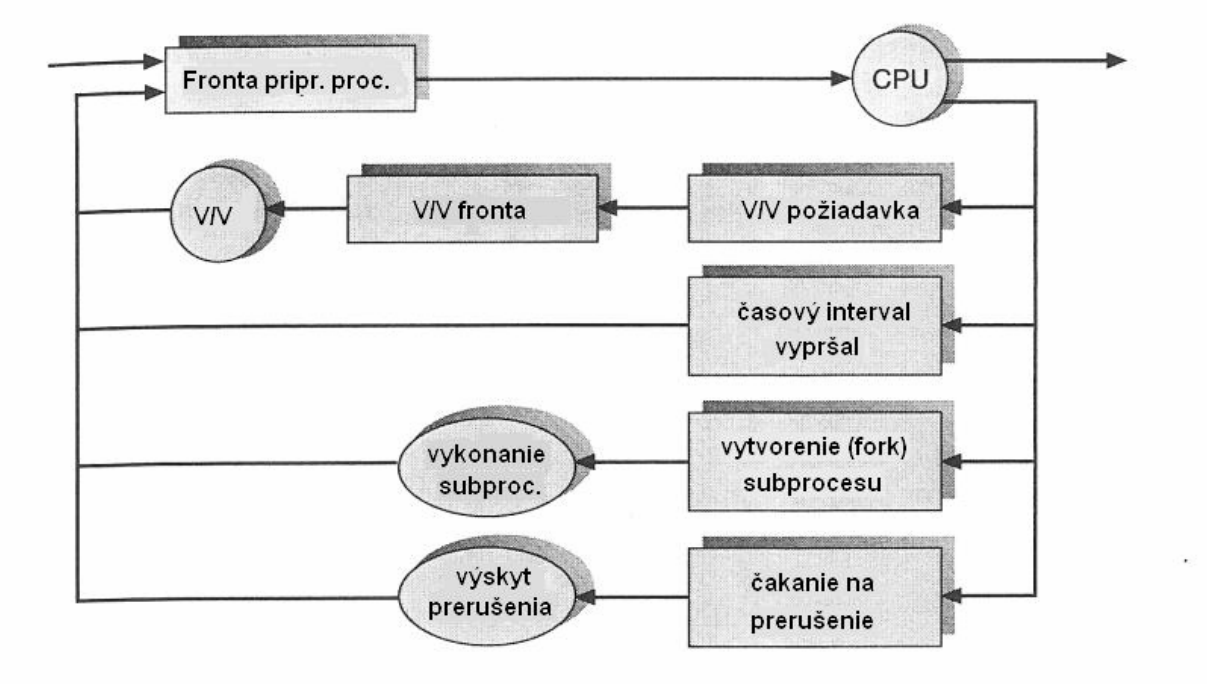
Ako procesy vstupujú do systému, sú ukladané do **frontu jobov** (*Job Queue*). Tento front pozostáva zo všetkých procesov v systéme. Procesy, ktoré sídlia v hlavnej pamäti a sú pripravené a čakajúce na vykonávanie, sú uložené v zozname nazývanom **front pripravených** (*Ready Queue*). Tento front je obyčajne uložený ako spojovací zoznam. Hlavička a päta frontu připravených obsahujú smerník na prvý a posledný PCB v zozname. Každý PCB rozšírime tak, aby obsahoval smerník, ktorý ukazuje na ďalší PCB vo fronte pripravených.

Operačný systém má tiež iné fronty. Keď je proces pridelený CPU, vykonáva sa chvíľu a eventuálne skončí, je prerušený alebo čaká na výskyt istej udalosti, ako napríklad dokončenie I/O žiadosti. V prípade I/O žiadosti, takáto žiadosť môže byť k zdieľanému zariadeniu, ako napríklad disku. Keďže systém má mnoho procesov, disk môže byť zaneprázdnený I/O požiadavkou onoho procesu. Proces preto môže čakať na disk. Zoznam procesov čakajúcich na určité I/O zariadenie sa nazýva **front zariadenia** (*Device Queue***)**. Každé zariadenie má

svoj vlastný front.



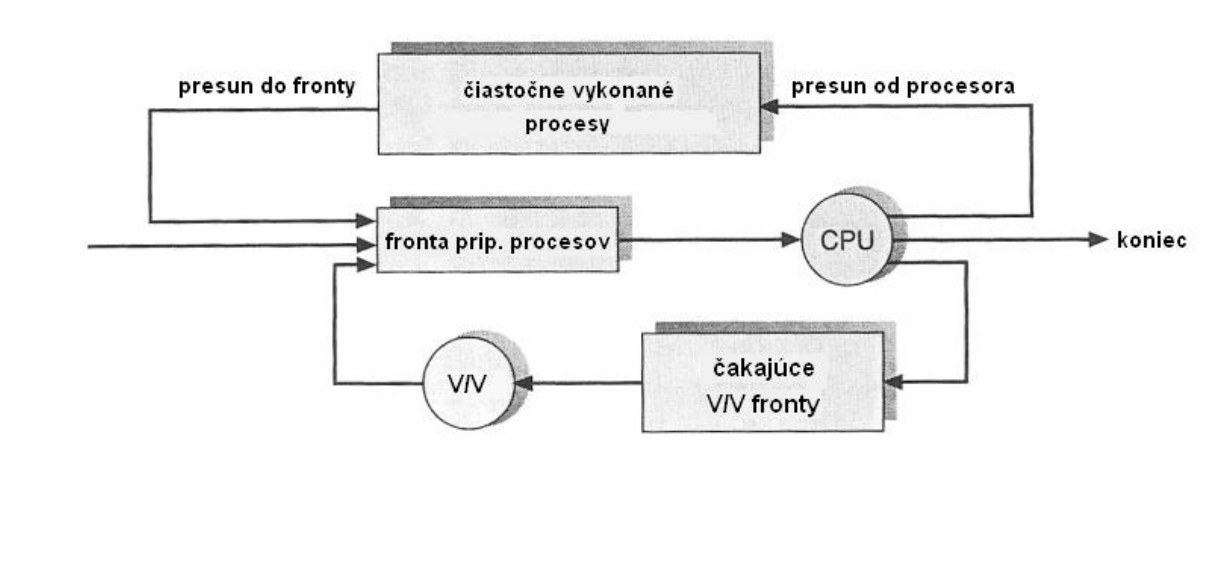
Zvyčajným znázornením rozvrhovania procesov je **diagram radenia do frontov.** Každý obdĺžnik reprezentuje front. Vidieť tu dva typy frontov: front pripravených a množinu frontov zariadení. Kružnice označujú zdroje systému, ktoré obsluhujú fronty a šípky označujú tok procesov v systéme.
Nový proces je na začiatku vložený do frontu pripravených procesov. Čaká v ňom, až kým nie je **vybraný na vykonanie** (*Dispatched*). Keď je proces pridelený k CPU a je vykonávaný, jedna z nasledujúcich udalostí môže nastať:
• Proces môže vydať I/O požiadavku a potom byť umiestnený v I/O fronte.
• Proces môže vytvoriť nový podproces a čakať na jeho ukončenie.
• Proces môže byť násilne odstránený od CPU ako výsledok prerušenia a položený späť do frontu pripravených procesov.
V prvých dvoch prípadoch proces sa potom prepne z čakajúceho stavu do pripraveného stavu a je vložený späť do frontu pripravených. Proces pokračuje v tomto cykle, až kým neskončí a potom je odstránený zo všetkých front a jeho PCB a všetky zdroje sú dealokované.



Rozvrhovanie

Počas svojho života proces migruje medzi rôznymi rozvrhovacími frontami. Operačný systém musí vyberať kvôli rozvrhovaniu procesy z týchto frontov nejakým spôsobom. Výber procesu je vykonaný na základe vhodného **rozvrhovacieho** programu.
V systémoch multiprogramovania často viac procesov vstúpi do systému, ako môžu byť vykonané okamžite. Tieto procesy sú nahraté (*Spooled*) na disk, kde sú držané na neskoršie vykonanie. **Dlhodobý plánovač** (*Job Scheduler*) vyberie proces z tohto súboru a zavedie ho do pamäte na vykonanie. **Krátkodobý plánovač** (*CPU Scheduler*) vyberie z viacerých procesov, ktoré sú pripravené v pamäti na vykonanie a priradí CPU jednému z nich.
Základný rozdiel medzi týmito dvoma plánovačmi je frekvencia ich vykonávania. Krátkodobé plánovanie musí vyberať nový proces pre CPU často. Proces sa môže vykonávať iba niekoľko málo milisekúnd predtým, ako bude čakať na I/O požiadavku. Krátkodobé plánovanie je často vykonávané najneskoršie každých 100 milisekúnd. Preto krátkodobé plánovanie musí byť rýchle. Ak pri krátkodobom plánovaní sa rozhoduje 10 milisekúnd o vykonaní procesu na 100 milisekúnd, tak potom 10/(100 + 10) = 9 percent z času CPU je premrhaných iba na plánovanie.
Na druhej strane, dlhodobé plánovanie je vykonávané menej často. Môže uplynúť aj viac minút medzi vytvorením nových procesov v systéme. Dlhodobé plánovanie riadi **stupeň multiprogramovania —** počet procesov v pamäti. Ak je stupeň multiprogramovania stály, potom priemerné množstvo novovytvorených procesov musí byť rovné priemernému počtu procesov, ktoré opúšťajú systém. Dlhodobé plánovanie môže byť vyvolávané, keď proces opúšťa systém. Takže, dlhodobý plánovač má viac času na výber procesu na vykonávanie. Dlhodobé plánovanie musí vyberať procesy starostlivo. Procesy sú väčšinou opísané ako I/O viazané alebo CPU viazané. **I/O viazané procesy** strávia viac času vykonávaním I/O, ako robením výpočtov. **CPU viazané procesy** generujú I/O požiadavky menej, viac času strávia robením výpočtov. Dlhodobý plánovač musí vybrať dobrý **mix procesov** z I/O viazaných a CPU viazaných. Ak všetky procesy sú I/O viazané, potom front pripravených procesov je vždy prázdny a CPU je nevyužité, takže krátkodobý plánovač bude mať málo roboty. Ak všetky procesy sú CPU viazané, potom I/O čakajúci front bude vždy prázdny, zariadenia budú

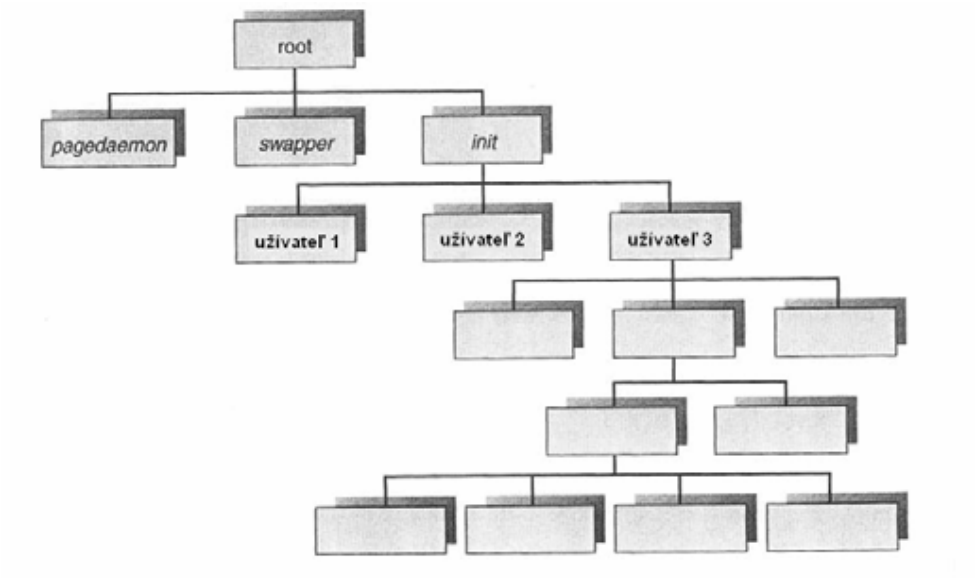
nevyužité a systém je zasa nevyvážený. Systém má najlepší výkon, keď bude mať správnu kombináciu CPU viazaných a I/O viazaných procesov.
V niektorých systémoch dlhodobé plánovanie môže chýbať alebo je minimálne. V systémoch zdieľania času často nie je žiadny dlhodobý plánovač, pretože jednoducho každý nový proces sa hneď položí do pamäte pre spracovanie krátkodobým plánovačom. Stabilita týchto systémov závisí tiež na počte dostupných terminálov, ale tiež od samo regulujúceho správania sa užívateľov. Ak výkonnosť systému klesne na neakceptovateľnú úroveň, niektorí užívatelia jednoducho skončia prácu.
Niektoré operačné systémy také, ako systémy zdieľania času, môžu zaviesť dodatočnú, strednú úroveň plánovania. Toto **stredne dobé plánomanie** odstraňuje procesy z pamäte na disk a tak redukuje stupeň multiprogramovania. O nejaký čas neskôr procesy môžu byť vložené späť do pamäte a ich vykonávanie môže pokračovať tam, kde skončilo. Tento systém sa volá **presun** (*Swapping*). Proces je presunutý von (z pamäte na disk) a neskôr je presunutý späť stredne dobým plánovačom.



Prepnutie kontextu
Prepínanie CPU na iný proces požaduje uložiť stav starého procesu a zaviesť uložený stav pre nový proces. Táto úloha sa nazýva **prepnutie kontextu** (*Context Switch*)**. Kontext** procesu je prezentovaný v PCB procesu; obsahuje hodnoty CPU registrov, stav procesu a informácie o riadení pamäti. Pri prepnutí kontextu, kernel uloží kontext starého procesu do jeho PCB a zavedie z PCB uložený kontext nového procesu, ktorý je naplánovaný na bežanie.

Operácie na procesoch
Procesy v systéme môžu byť vykonávané súčasne a musia byť vytvárané a odstraňované dynamicky. Preto operačný systém musí poskytovať mechanizmus na vytváranie a ukončenie procesov.

Vytváranie procesov
Proces môže vytvoriť niekoľko nových procesov pomocou systémového volania `create-process`. Vytvárajúci proces sa nazýva rodičovský proces, zatiaľ čo nové procesy sa nazývajú deti tohto procesu. Každý z týchto nových procesov môže vytvoriť nové procesy a tak sa formuje *strom* procesov



Procesy vo všeobecnosti potrebujú isté zdroje na vykonanie svojej úlohy. Detský podproces môže byť schopný získať zdroje priamo z operačného systému alebo je obmedzený iba na množinu zdrojov rodičovského procesu. Rodič môže rozdeliť svoje zdroje medzi svoje deti alebo môže zdieľať niektoré zdroje (ako pamäť alebo súbory) medzi niektoré zo svojich detí. Toto obmedzenie detských procesov, na využívanie iba podmnožiny rodičovských zdrojov zabezpečuje, aby procesy nemohli zahltiť systém vytváraním veľkého množstva podprocesov. Pri vytvorení procesu tento dostane tiež inicializačné dáta (alebo vstup), ktorý mu môže byť odovzdaný rodičovským procesom. Keď proces vytvorí nový proces, dve možnosti existujú ohľadne vykonávania:
1. Rodič pokračuje vykonávanie súbežne s dieťaťom.
2. Rodič čaká, až kým niektoré alebo všetky jeho deti neukončia.
Sú tiež dve možnosti adresového priestoru pre nový proces:
1. Detský proces je duplikát rodičovského procesu.
2. Detský proces má svoj program zavedený do svojho adresového priestoru.
Na ilustráciu týchto rôznych implementácií, pozrime sa na UNIX, kde každý proces je identifikovaný pomocou **identifikátora procesu** (*Process Identifier — PID*), ktorým je jedinečné celé číslo. Nový proces je vytvorený pomocou systémového volania `fork`. Nový proces pozostáva z kópie adresového priestoru rodičovského procesu. Tento mechanizmus umožňuje rodičovskému procesu jednoducho komunikovať s detským procesom. Oba procesy (rodič aj dieťa) pokračujú vo vykonávaní v inštrukcii za systémovým volaním `fork` len s jednou výnimkou: návratová hodnota pre systémové volanie `fork` je nula pre nový, detský proces, zatiaľ čo (nenulový) identifikátor procesu dieťaťa je vrátený rodičovi.
Po ukončení systémového volania `fork` je typicky použité systémové volanie `execlp`, volané jedným z dvoch procesov, aby nahradilo pamäťový priestor procesu kódom nového programu. Systémové volanie `execlp` zavedie do pamäti binárny súbor — zničí obraz paměti programu, ktorý obsahuje systémové volanie `execlp` — a začne jeho vykonávanie. Pri tejto metóde dva procesy sú schopné komunikovať a potom ísť vlastnou cestou. Rodič môže potom vytvoriť viac detí alebo, ak už nemá čo robiť, pokiaľ deti bežia, môže vydať systémové volanie `wait,` ktoré ho odstráni z frontu pripravených procesov a presunie do frontu čakajúcich, až kým dieťa neskončí.

Nasledujúci program ilustruje predošlé systémové volania v UNIX. Rodič vytvorí detský proces použitím systémového volania `fork`. Teraz máme dva rozdielne procesy bežiace s kopkou rovnakého programu. Hodnota `pid` pre detský proces je nulová; pre rodiča je to celočíselná hodnota väčšia ako nula. Detský proces prekrýva svoj adresový priestor s UNIX príkazom `/bin/ls` (používaným na výpis obsahu

adresára) použitím systémového volania `execlp`. Rodič čaká na ukončenie detského procesu pomocou systémového volania `wait`. Keď detský proces skončí, rodičovský proces pokračuje z miesta, odkiaľ bolo zavolané systémové volanie `wait`.

```
#include <stdio.h>
        void main( ) {
                int pid;
                pid = fork();           /* vytvor iný proces */
                if (pid < 0) {          /* chyba sa vyskytla  pri vytvorení nového procesu*/
                        fprintf(stderr, "Fork sa vykonal chybne");
                        exit(-1);       /* návrat do operačného systému */
                } else
                        if (pid == 0)           /* detský proces */
                                execlp("/bin/ls","ls",NULL);    /*zavedieme program ls do detského procesu*/
                        else {    /* rodičovský proces  */
                                wait(NULL);   /* rodič počká na ukončenie dieťaťa */
                                printf("Dieta ukončené");
                                exit(0);        /* návrat do operačného systému */
                        }
        }
```

- netreba vediet ☺

Ukončenie procesu

Proces skončí, keď vykoná svoju finálnu inštrukciu a požiada operačný systém o svoje odstránenie použitím systémového volania `exit`. V tomto bode proces môže vrátiť výstupné hodnoty rodičovskému procesu (pomocou systémového volania `wait`). Všetky zdroje procesu — vrátane fyzickej a virtuálnej pamäte, otvorené súbory a I/O bufre — sú uvoľnené operačným systémom. Ukončenie procesu môže nastať aj za iných okolností. Proces môže ukončiť iný proces vhodným systémovým volaním (napríklad `abort`). Zvyčajne iba rodič procesu môže volať toto systémové volanie. Inak by užívatelia mohli zrušiť ľubovoľný proces. Rodič preto potrebuje poznať identity svojich detí. Preto, keď jeden proces vytvorí nový proces, identita novovytvoreného procesu (*pid*) je odovzdaná rodičovi.
Rodič môže ukončiť vykonávanie jedného zo svojich detí z rôznych dôvodov ako sú:
• Dieťa prekročí hranicu niektorých zdrojov, ktoré má pridelené. To vyžaduje mechanizmus, ktorým rodič bude kontrolovať stav svojho dieťaťa.
• Úloha priradená dieťaťu nie je ďalej požadovaná.
• Rodič končí a operačný systém nedovoľuje dieťaťu pokračovať, ak jeho rodič ukončí. Na takom systéme, ak proces skončí (buď normálne alebo abnormálne), potom všetky jeho deti musia tiež byť ukončené. Tento fenomén, nazývaný ako **kaskádové ukončovanie,** je zvyčajne inicializovaný operačným systémom.
Na ilustráciu vykonávania a ukončovania procesu uvažujme, ako sa to robí v operačnom systéme UNIX. Proces môžeme ukončiť použitím systémového volania `exit`. Rodičovský proces môže počkať na ukončenie detského procesu použitím systémového volania `wait`. Systémové volanie `wait` vracia identifikátor (*pid*) ukončeného dieťaťa, takže rodič vie, ktoré z jeho možných detí ukončilo. Ak rodič skončí, všetkým deťom sa priradí ako ich nový rodič proces `init`. Takže deti stále majú rodiča, ktorý zhromažďuje ich stav a štatistiky vykonávania.

Zhrnutie

Proces je program vo vykonávaní. Ako sa proces vykonáva, on mení stav. Stav procesu je definovaný aktuálnou aktivitou procesu. Každý proces môže byť v jednom z nasledujúcich stavov: nový, pripravený, bežiaci, čakajúci, alebo ukončený. Každý proces je reprezentovaný v operačním systéme svojim vlastným PCB (*Process Control Block*).
Proces, keď nie je vykonávaný, tak je uložený do frontu čakajúcich. Dva základné fronty v operačnom systéme sú front I/O požiadaviek a front pripravených. Front pripravených obsahuje všetky procesy, ktoré sú pripravené na vykonávanie a čakajú na CPU. Každý proces je reprezentovaný PCB a PCB môžu byť spájané dohromady k vytvoreniu frontu. Dlhodobé (alebo job) plánovanie je výber procesov, ktorým je umožnené súťažiť o CPU. Dlhodobé plánovanie je silne ovplyvnené alokáciou zdrojov, predovšetkým pamäte. Krátkodobé (alebo CPU) plánovanie je výber jedného procesu z frontu pripravených.
Procesy v systéme sa môžu vykonávať súbežne (*Concurrently*). Je niekoľko dôvodov pre umožnenie súbežného vykonávania procesov: zdieľanie informácii, zvýšenie rýchlosti výpočtu, modularita a pohodlie. Súbežné vykonávanie potrebuje mechanizmy pre vytvorenie procesu a jeho odstránenie.

Zdroj: M. Schmotzer: Operačné systémy
J. Studenovský: Operačné systémy - Proces