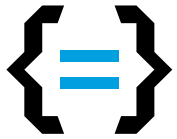


Univerzita Hradec Králové
Fakulta informatiky a managementu

Procesy a vlákna

Mgr. Josef Jan Horálek, Ph.D. & Ing. Tomáš Svoboda, Ph.D.

- Základním úkolem jádra je správa běžících procesů a vláken:
  - vytváření
  - plánování
  - nastavování
  - ukončování

- Proces – běžící program
  - v OS Linux jde o objekt, který pracuje podle kódu program (včetně knihoven), využívá svůj vlastní adresní paměťový prostor. Využívá služby jádra a komunikuje různými způsoby s ostatními procesy
- Vlákno
  - objekt pracující podle kódu programu, liší se od procesu tím, že je jeho součástí a sdílí prostředky (adresní prostor paměti, prostředky jádra atd.) s ostatními vlákny procesu

Proces, vlákno, úloha

- Úloha (task) – objekt, jehož kód se sekvenčně vykonává.
- Každému vláknu procesu odpovídá jedna úloha (kromě toho může proces využívat ještě další, navenek neviditelné, úlohy přímo v jádře – např. vlákna jádra spravující pracovní fronty.

- **Na jednom jádře procesoru běží v určitý okamžik nejvýše jedna úloha.**

Plánování úloh

= Úlohy běží na jednotlivých procesorech v rámci přidělených časových kvant. Úloha tedy dostane časové kvantum, a dokud ho nevyčerpá, a není ji z nějakého důvodu procesor odebrán plánovačem, vykonává své programové instrukce.

Plánování úloh

- Jak bude plánování vypadat ovlivňují tři faktory:
  - použitý plánovač,
  - priorita úlohy,
  - vlastnosti jádra (preemptivita),
  - každé vlákno procesu je z hlediska plánování chápáno jako samostatná úloha, je tedy plánováno nezávisle na jiných vláknech téhož procesu.

Plánovače

- Plánovač má nejzásadnější vliv na plánování. V jádře jsou dnes implementovány čtyři plánovače.
  - normální plánovač – pro většinu použití,
  - dávkový plánovač – jako normální plánovač, ale s úpravou pro neinteraktivní úlohy,
  - plánovač FIFO – pro úlohy zpracovávané v reálném

Normální plánovač

- Tento plánovač je použit ve výchozím případě.
  - využívá jedinou úroveň statické priority, místo toho používá hodnoty nice (40 úrovní) a časová kvanta přiděluje na základě dynamických priorit, které jsou určovány podle toho, jak úloha v minulosti využívala procesor.
- Časové kvantum tak získá vždy úloha, která je první ve frontě, velikost kvanta odpovídá dynamické prioritě.

Dávkový plánovač

- Vhodný pro úlohy, které mají velké požadavky na čas procesoru a nepotřebují být interaktivní.
  - liší se výpočtem dynamických priorit, na základě které se určuje velikost časového kvanta.
- Úloha se posuzuje vždy tak, jako kdyby neustále požadovala běh na procesoru, je tedy permanentně penalizována.

Plánovač FIFO

- Tento plánovač funguje tak, že úloze přidělí procesor, a dokud ho on sám neodevzdá, žádná jiná úloha na tomto procesoru příležitost k běhu nedostane.
- Využívá 99 úrovní statických priorit. Procesor je přidělen úloze, která má vyšší prioritu.

Plánovač round-robin

- Podobný jako plánovač FIFO, ale liší se přidělováním časových kvant, které přiděluje dokola na určité úrovni priority (úrovní 99).
- Úloha tedy běží nejdéle po dobu přiděleného časového kvanta, pak jí je procesor odebrán a je předán další úloze se stejnou prioritou.

Priority běhu

- Hodnota nice je priorita s obráceným významem
  - čím vyšší nice, tím nižší priorita. Výchozí hodnotou je nula, záporné hodnoty (do -20 znamenají větší prioritu, kladné (do 19) nižší,
  - úloha může vždy snížit svoji prioritu (zvýšit NE).
- Statická priorita – nabývá hodnoty od 1 do 99. Čím vyšší hodnota, tím vyšší priorita.
  - nice -12 large-job ;úloha běží s nížší prioritou
  - nice --12 large-job ;vyšší priorita

Preemtivita jádra

- Jádro můžeme zkompilovat třemi způsoby:
- Nepreemtivní chování – úloha se v jádře musí sama vzdát procesoru.
- Dobrovolně preemtivní chování – jsou přidány body, kde se úloha vzdá procesoru.
- Plně preemtivní chování – plánovač úloze odebere procesor, pokud vyčerpala časové kvantum.

Proces a jeho vlastnosti

- Každý proces má jednoznačnou číselnou identifikaci označenou zkratkou PID (process idenfifiter). Tento identifikátor je jedinečný, po použití se ale recykluje a bude co nejdříve použit znovu.
- Každý proces má svého rodiče – proces kterým byl vytvořen.
- Jedinou výjimkou je proces init.

- Problematika paměťového prostoru:
  - stránky namapované procesem se z hlediska procesu zkopírují – ne však hned,
  - dokud se ze stránek jen čte, zůstávají společné pro starý i nový proces,
  - zkopírování nastává až v okamžiku, kdy se některý z procesů pokusí o zápis nebo si zkopírování explicitně vynutí.
- Souborové deskriptory:
  - až a výjimky se dědí – jsou tedy platné beze změny i v novém procesu,
  - problém hrozí v nechtěném přístupu k datům a také výrazné bezpečnostní riziko,
  - proti nechtěnému úniku deskriptorů se používá uzavření všech možných deskriptorů hned po jeho zkopírování, krom těch které potomek zdědit má.

- Oprávnění procesů:
  - reálná – odpovídají oprávnění uživatele, který spustil program příslušející k procesu,
  - efektivní – mohou být ovlivněna tím, že má program nastaven bit SUID, nebo-li proces běží s právy vlastníka souboru s programem.
- Při vytváření nového procesu se toto všechno opět dědí.
- Po skončení procesu se nastaví jeho návratová hodnota, kterou si může zkontrolovat rodičovský proces a podle ní zjistit, zda byl úspěšný atd.

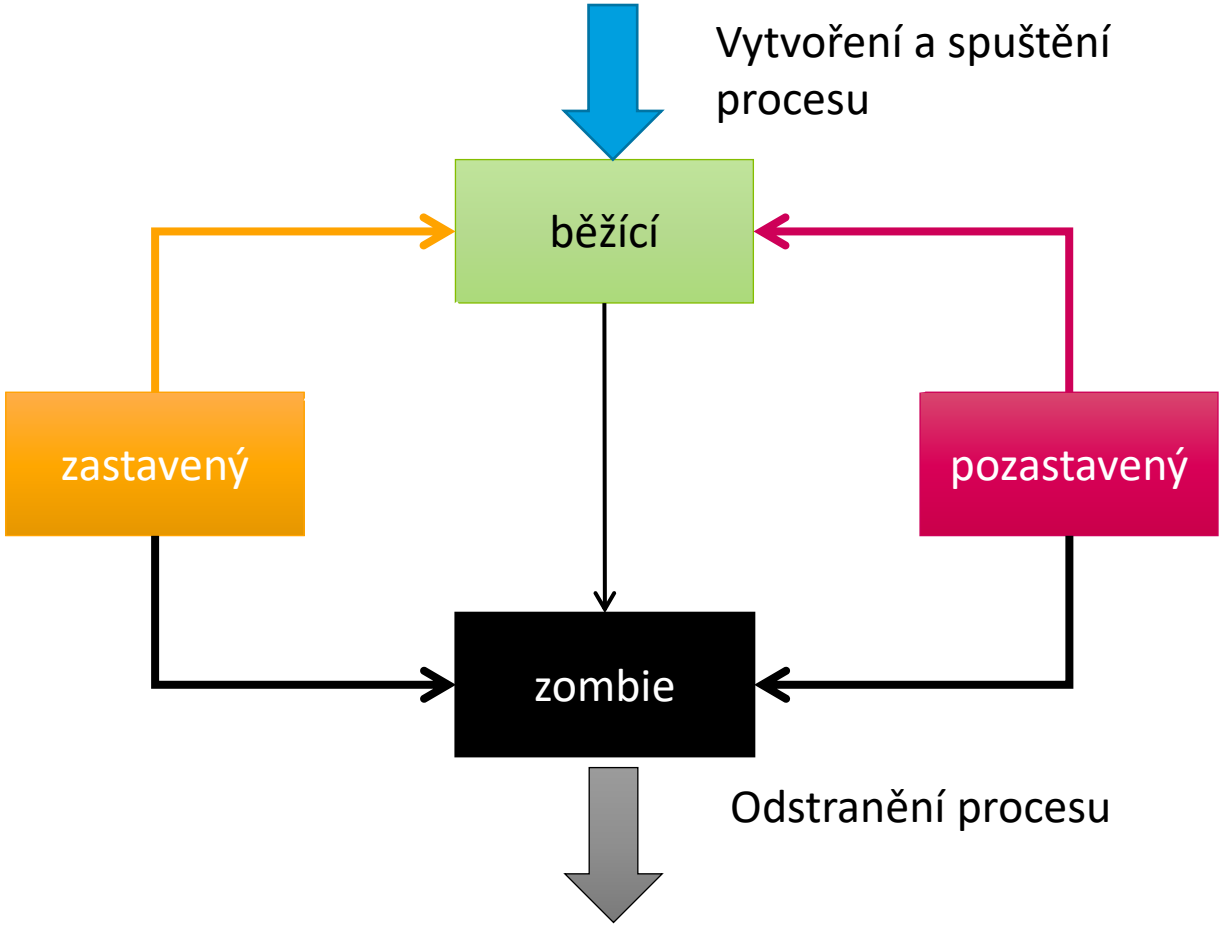
Životní cyklus procesu

- Běžící – procesory vykonávají programový kód procesu nebo některá vlákna čekají na splnění nějaké podmínky.
- Zastavený – proces obdržel signál SIGSTOP,SIGTTOU, SIGTTIN, SIGTSTP, a byl tedy zastaven. Žádný z procesorů nevykonává kód tohoto procesu.
- Pozastavený – provádění kódu bylo přerušeno (signálem SIGTRAP) kvůli splnění nějaké podmínky, například definované debuggerem. Tento stav je velmi podobný stavu zastavený.
- Zombie – provádění kódu bylo dokončeno, již neběží na žádném procesoru. Proces zůstává v tomto stavu, dokud není nadřazeným procesem jádro požádáno o jeho odstranění.

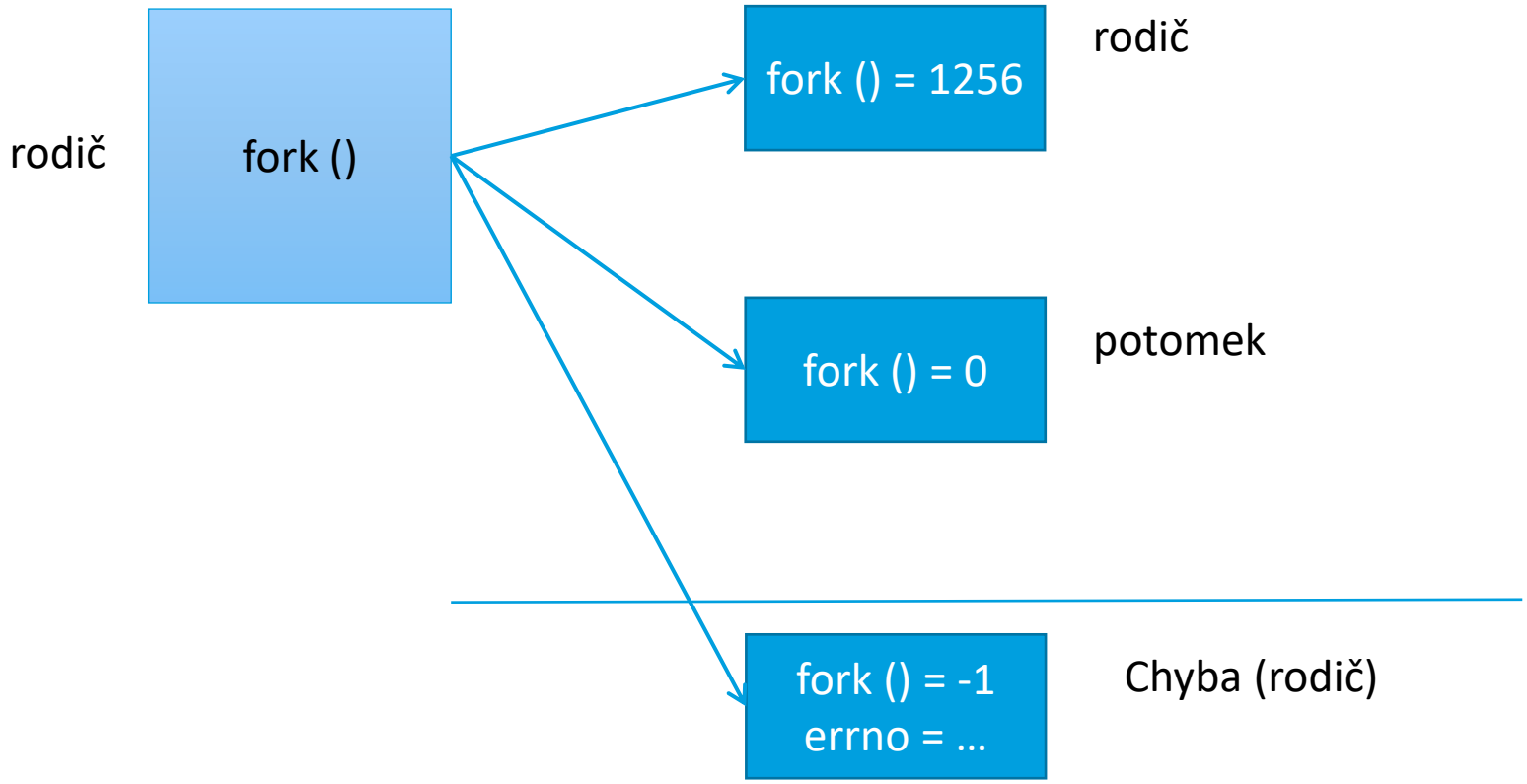
Životní cyklus procesu

- Proces tedy za normálních okolností běží dokud má k dispozici instrukce, které lze vykonat. Po dokončení běhu procesu (signál SIGCHLD) může mít rodičovský proces nataveno SIG_IGN – pak proces tiše zanikne.
- Pokud má však nastavenou explicitní obsluhu, proces nezanikne a zůstává ve stavu zombie.

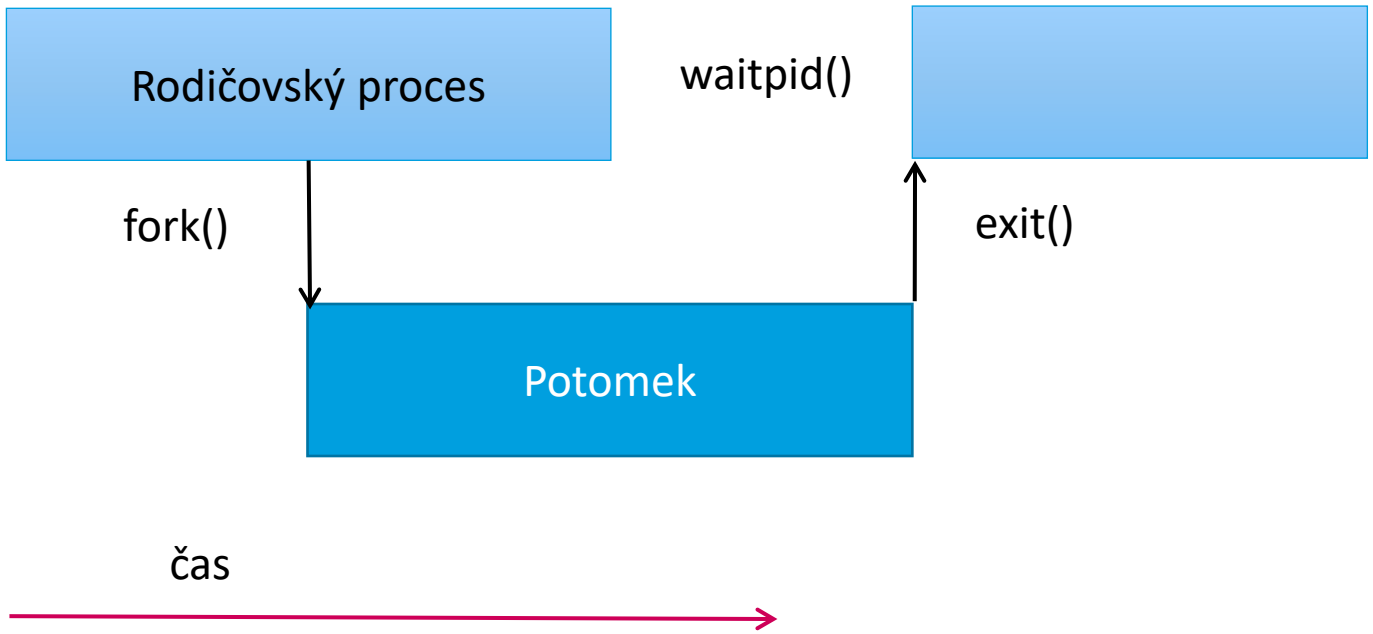
Životní cyklus procesu



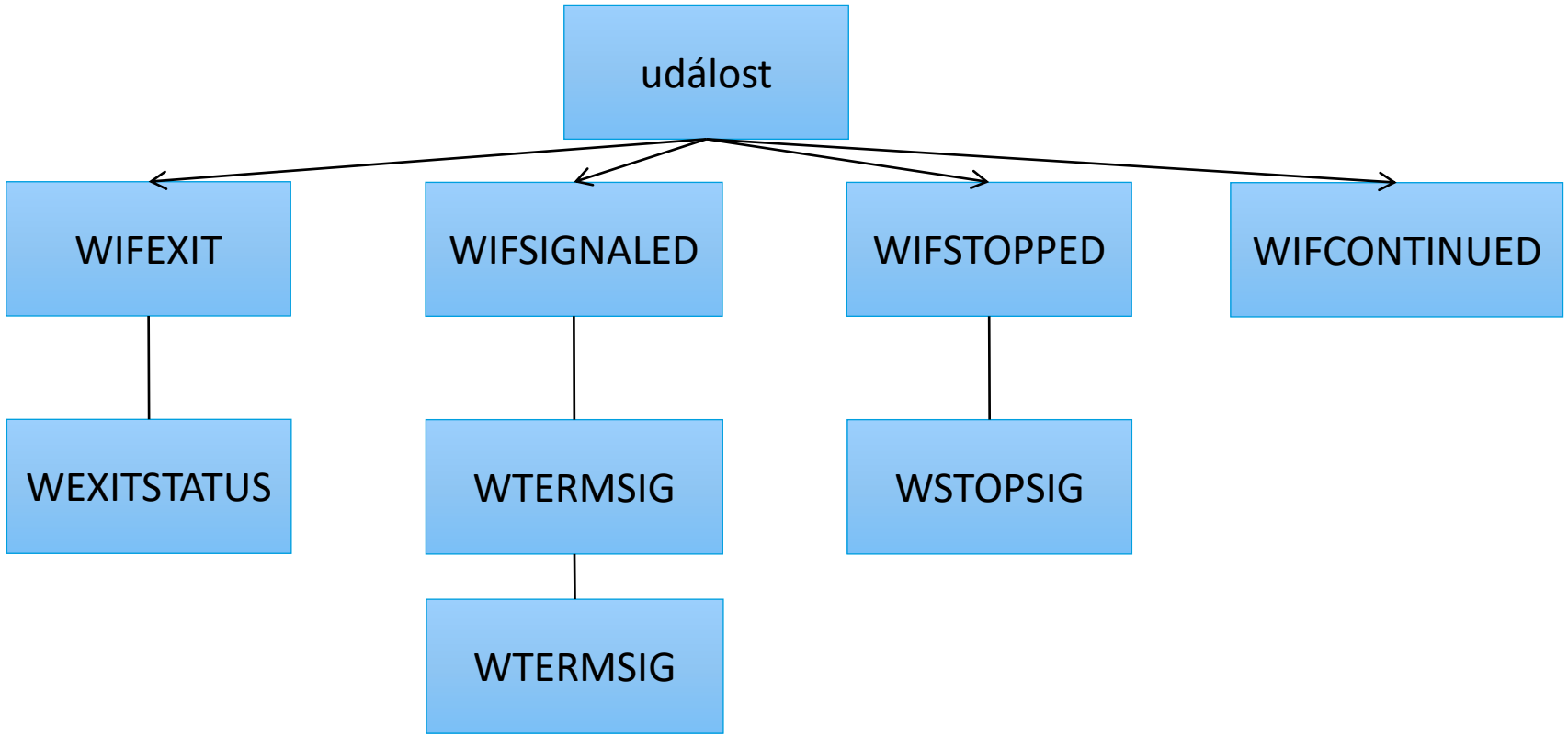
Implementace správy procesu



Implementace správy procesu



Implementace správy procesu



Proces skončil

Neobsloužený signál

Proces zastaven

Proces pokračuje

Práce s vlákny

- Hlavní vlákno – vlákno, které je spuštěno v procesu jako první – nepotřebuje explicitně žádnou obsluhu.
- Každé vlákno označeno identifikátorem TID (thread identifier) přidělované ze stejného číselného prostoru jako PID.
- TID hlavního vlákna = PID procesu.
- Pro zjištění TID voláme funkcí gettid().

Životní cyklus vlákna

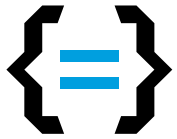
- Běžící – nějaký procesor právě vykonává kód vlákna.
- Připravené k běhu – vlákno čeká na časové kvantum pro svůj běh.
- Přerušitelně uspané – vlákno čeká na splnění nějaké podmínky – lze ho probudit, takže např. může obsloužit signál.
- Nepřerušitelně uspané – vlákno čeká na splnění podmínky, ale nelze ho probudit – nastává zřídka a to většinou jen pro vlákna v procesu.
- Zastavené – vlákno bylo zastaveno signálem.
- Pozastavené – opět podobná situace jako u procesu.
- Skončené – vlákno doběhlo do konce a čeká se, až se s ním vlákno spojí.

Základní práce s vlákny

- S vlákny se obvykle pracuje prostřednictvím knihovny libpthread, která hlavně implementuje rozhraní POSIX Threads.
- Knihovna poskytuje vše potřebné prostřednictvím hlavičkového souboru pthread.h
- Knihovna pthread je v uživatelském prostoru a většina vláken se vykonává v jejím rámci.
- Běh vlákna může skončit trojím způsobem:
  - Doběhnutí startovací funkce (vrácení hodnoty),
  - Zavolání funkce pthread_exit(),
  - Zrušení vlákna.

Ukončování a úklid

- Speciální možnosti ukončení:
  - odloučení vlákna – nechceme-li na vlákno čekat, lze ho převést do odloučeného stavu pthread_detach() – skončí-li takové vlákno automaticky po sobě uklidí a uvolní všechny související prostředky.
  - zrušení vlákna – můžeme jej zrušit synchronně nebo asynchronně.
- Důležitou funkcí při ukončování vlákna je zavolat tzv. úklidovou obslužnou funkci (clean-up handler).
- Tato funkce je volána ve chvíli, když někdo zvenku zruší vlákno, a cílem je uklidit data před úplným ukončením vlákna.



Univerzita Hradec Králové
Fakulta informatiky a managementu

Děkuji za pozornost

Další téma: Spouštění programů