Univerzita Hradec Králové
Fakulta informatiky a managementu

Správa procesů a vláken

Mgr. Josef Horálek

FIM UHK

Správa procesů a vláken

- Jeden z nejdůležitějších úkolů jádra operačního systému
- Linuxové jádro dosáhlo důkladné implementace umožňující například plnohodnotné použití vláken až v řadě 2.6

FIM UHK

Úlohy, procesy, vlákna

- V pojetí linuxového jádra do značné míry platí proces = vlákno
- Původně vlákna nepodporována – proces vždy jednovláknový
- Dnes máme
  - plnohodnotné procesy x odlehčené procesy
- Úloha – entita plánovaná pro běh procesoru
- Proces – všechno co vykonává kód jednoho spustitelného souboru a sdílí jeden adresní prostor
- Vlákno – úlohy uvnitř procesu

FIM UHK

Vlastnosti a stav úlohy

- **Task_struct** – datová struktura popisující úlohu
- Obsahuje obrovské množství položek, které se dynamicky mění
- Důležité položky:
- Stav úlohy
  - Příznaky úlohy
  - Datové struktury

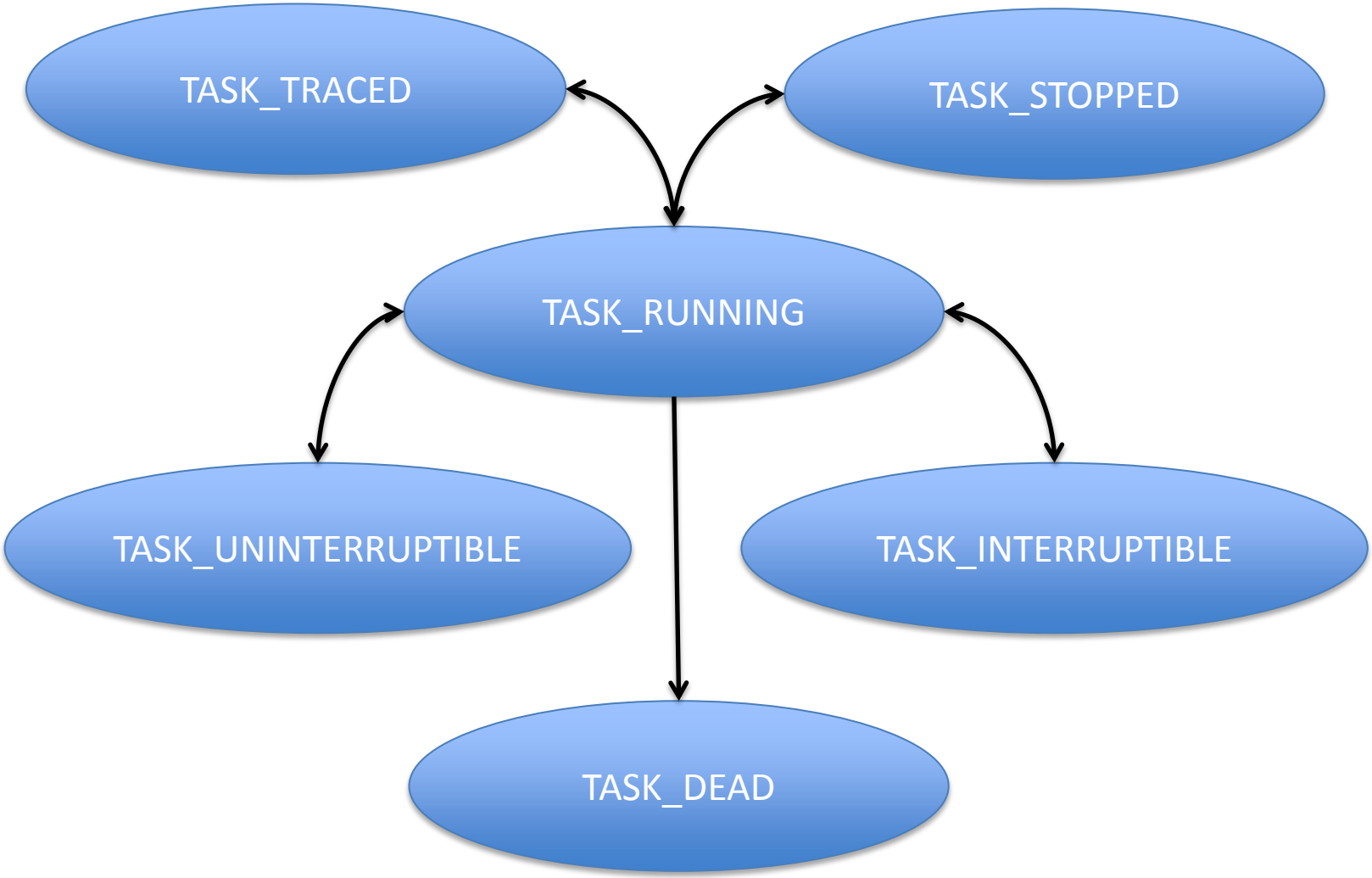
FIM UHK

Stav úlohy

- Položka state reprezentuje stav úlohy
- Nastavuje se pomocí funkce **set_task_state()** a **set_current_state()**
- Hodnoty proměnné state:
  - TASK_RUNNING
  - TASK_INTERRUPTIBLE
  - TASK_UNITERRUPTIBLE
  - TASK_STOPPED
  - TASK_TRACED
  - TASK_DEAD

FIM UHK

Stav úlohy



FIM UHK

další datové

- Příznaky – parametr úlohy uložený v položce **flags** struktury **task_struct**
- Jedná se o bitovou masku a příznaky se tak mohou kombinovat
  - PF_STARTING
  - PF_EXITING
  - PF_FORKNOEXEC
  - PF_SIGNALED
  - PF_MEMALLOC
  - PF_RANDOMIZE
  - PF_MEMPOLICY

FIM UHK

další datové

- Dalším důležitým parametrem je ukazatel na strukturu **thread_info**
- Každá architektura má vlastní definici – jedná se o data silně spjatá s HW
- Obsahuje informace o aktuální procesoru úlohy, o návratu ze systémového volání, o adresném prostoru atd.

FIM UHK

Příznaky úlohy, další datové struktury

- **PID** – datová struktura sloužící k jednoduché identifikaci vlákna, procesu, skupiny procesů a session
- V porovnání s TID, TGID/PID, PGID, SID má východu, že je vždy jednoznačná a netrpí problémy s recyklací číselných hodnot
- Ukládá se do hashové tabulky, kde je lze velmi rychle najít podle číselné hodnoty

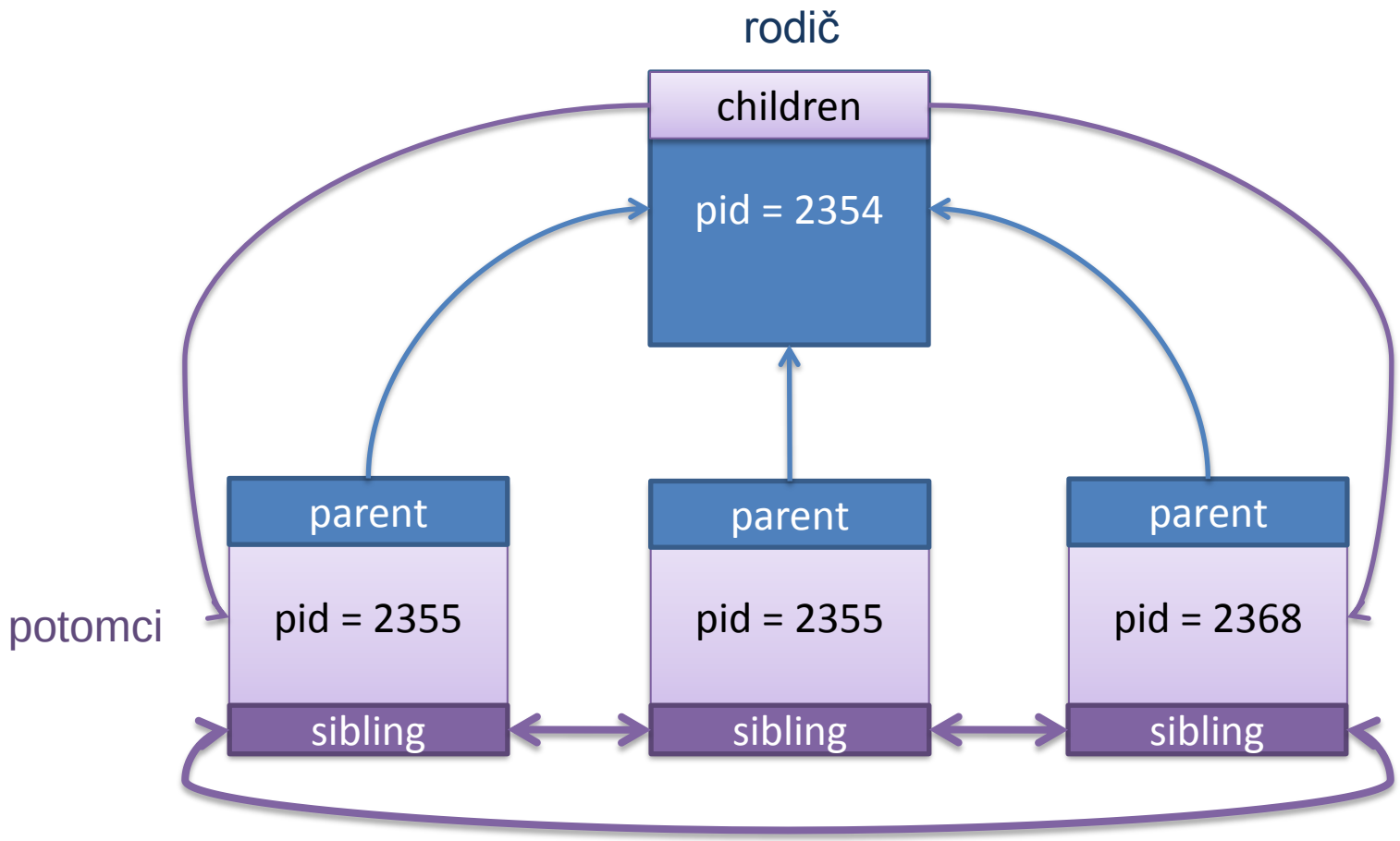
FIM UHK

Vztahy mezi úlohami

- Mezi jednotlivými úlohami panují určité vztahy
- Struktura **task_struck** obsahuje položky, které tyto vztahy popisují
  - Parent
  - Real_parent
  - Children
  - Sibling
  - Group_leader
  - Ptrace_children
  - Ptrace_list
  - Tgid
  - Signal->pgrp
  - Signal->session

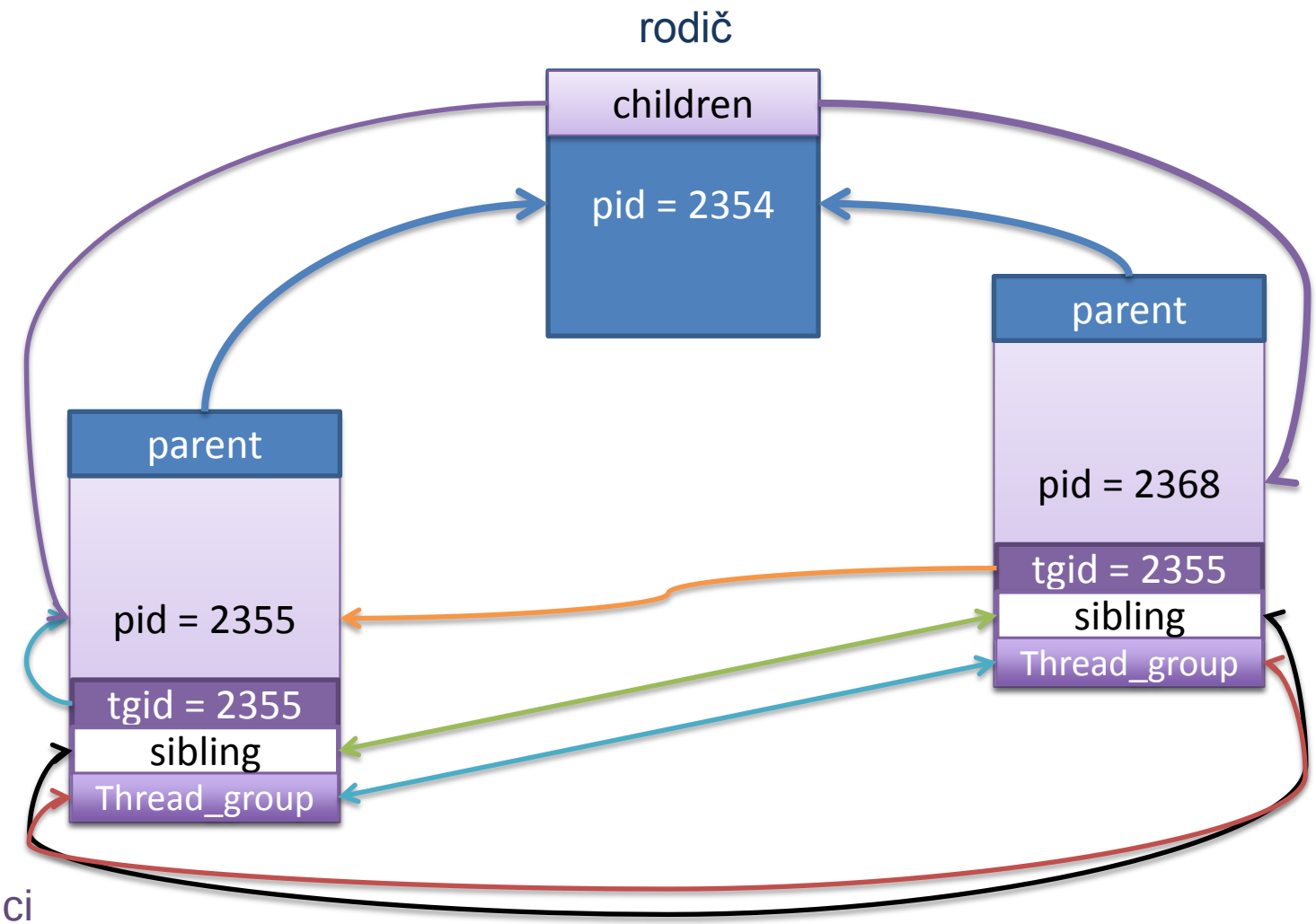
FIM UHK

Procesy – rodič a potomci



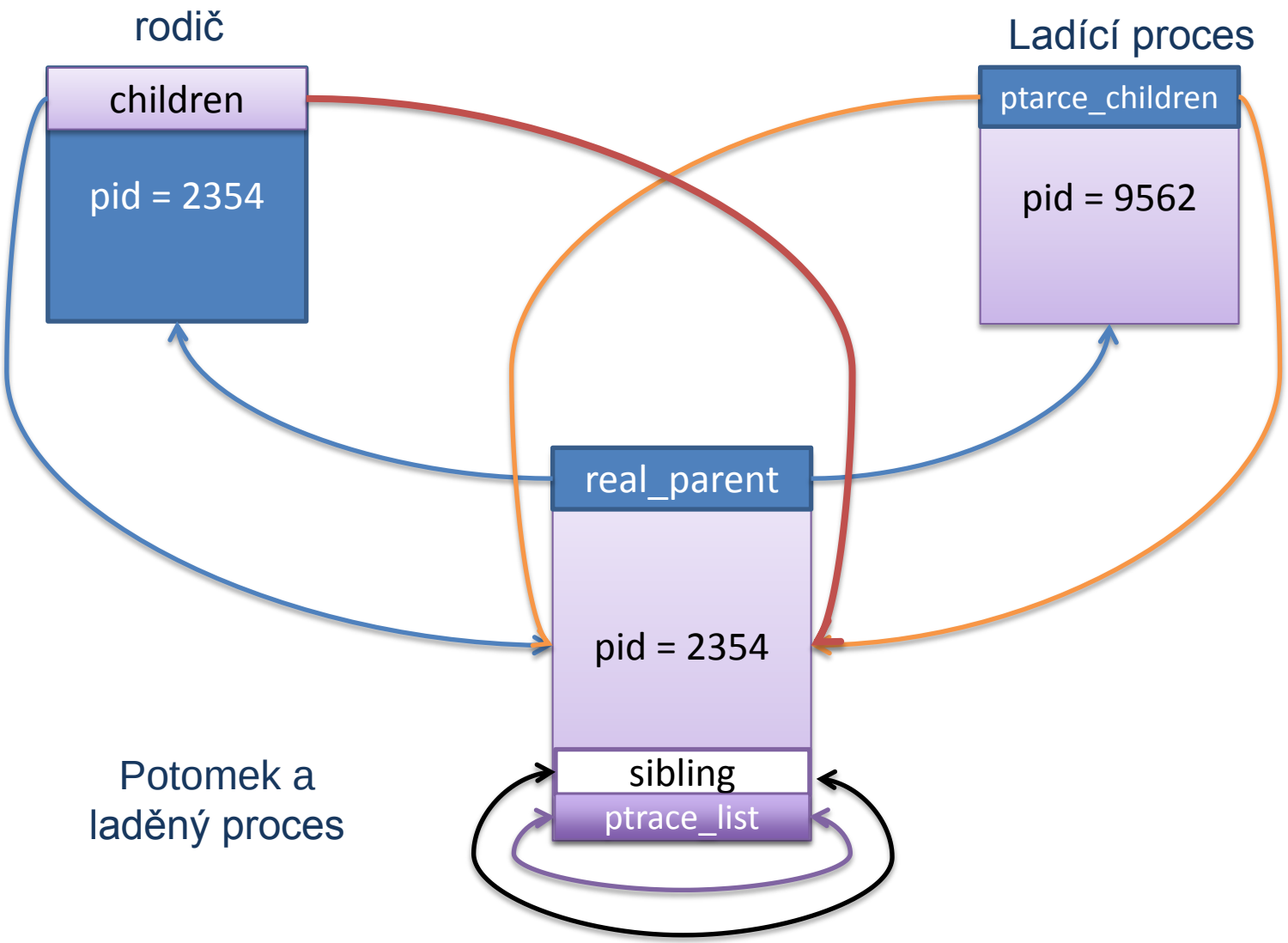
FIM UHK

Procesy – rodič a jeho vícevláknový potomek



FIM UHK

Procesy – ladění pomocí ptrace()



FIM UHK

Vytváření a rušení úloh

- Vytvoření úlohy – vždy centrální
- Hlavní výkonnou funkcí je **do_fork()** zde končí volání **fork()**, **clone()**a **kernel_thread()**
- **do_fork** začne alokací struktury pid – nastaví případný ladící příznak a provede kopírovací práce

FIM UHK

Kopírování úlohy

- Ve starších unixových systémech vytvoření procesu = kompletní zkopírování procesu
- V Linuxu NE
  - paměť se kopíruje až při zápisu do stránky
  - lze určit co kopírovat
  - lze určit co sdílet

FIM UHK

Kopírování úlohy

- Proces kopírování zajišťuje funkce **copy_process()**, která provede:
- Kontrolu příznaků
- Ověří oprávnění
- Zkopíruje základní strukturu úlohy
- Kontrola limitu
- Zvýšení reference
- Aktualizace dalších stavových hodnot
- Do struktury úlohy se nastaví příznaky a struktura pid

FIM UHK

Kopírování úlohy

- Dále se kopíruje vše co má nastaveno příznak pro zkopírování – souborové deskriptory, signály a jejich obsluha, paměťové struktury atd.
- Vždy se volají všechny funkce, teprve uvnitř každé se rozhoduje, jak a zda se bude kopírovat
- Kopírování se dokončí funkcí **copy_thread()**

FIM UHK

Kopírování úlohy

- Poté pokračuje inicializace datových položek, následuje mapování úlohy na procesor, nastavení procesoru a masky procesů
- Pak dojde k přepočítání signálů, aby byli doručeny určité relaci nebo skupině procesů
- A následuje řada nastavení, které definují vtahy mezi rodiči, skupinou procesů, session atd.a aktualizují se stavové údaje

FIM UHK

Ukončení běhu úlohy

- Implementace ukončení běhu úlohy je veslice složitá záležitost
- Běh končí vždy stejným mechanismem
- Na nejvyšší úrovni využívá funkci **do_group_exit()**, která je volána systémovým voláním **exit_group()** a při neobsloužených signálech
- Ta zajistí ukončení všech vláken ve skupině daného procesu – posílá všem signál **SIGKILL** a pak zavolá **do_exit()**

FIM UHK

Ukončení běhu úlohy

- **do_exit()** začíná kontrolami – posílá notifikaci a aktualizuje paměť systému
- Po posledním vlákně ve skupině ruší časovače a odstraní jejich infrastrukturu
- Postupně uvolňuje jednotlivé komponenty (paměť, souborové deskriptory, souborové systémy..)

FIM UHK

Ukončení běhu úlohy

- Dalším krokem jsou notifikace
- Nejdříve se posílá zpráva přes connector a volá se **exit_notify()**, která mimo jiné zajistí přesměrování čekajících signálů do jiných vláken, potomky předá procesu **init**
- Změní ukončovací stav úlohy na **EXIT_ZOMBIE** a pokud nikdo nečeká pak přímo na **EXIT_DEAD**

FIM UHK

Ukončení běhu úlohy

- Uvolňovací funkcí **release_task()**se ukončuje daná úloha a také se zasílá signálová notifikace vláknu, které odstartovalo ukončení skupiny vláken a je ve stavu zombie
- **sched_exit()** zajistí dostatečné časové kvantum na zbytek ukončovacích prací

FIM UHK

Ukončení běhu úlohy

- Na samém závěru ukončování se úloha ještě jednou naplánuje – volá se funkce **finish_task_switch()** s voláním **put_task_struct()** , které dekrementuje počítadlo referencí – pokud na úlohu již nikdo nečeká klesne na nulu a úloha se odstraní
- Nedostane již časové kvantum a zůstane uzavřena

FIM UHK

Ukončení běhu úlohy

- Pokud na úlohu čeká její rodič, provede se definitivní uvolnění ve funkci **wait_task_zombie()**. Poté co se v této funkci změní stav úlohy na **EXIT_DEAD** a přečtou se potřebné údaje o úloze.

FIM UHK

Vlákna/procesy jádra

- Speciálním druhem vlákna je vlákno jádra (kernel thread) někdy se mluví i o procesu jádra
- Jde v podstatě o totéž, jelikož běží v adresním prostoru jádra společném pro všechny vlákna jádra
- Za vlákno jádra se považujeme jediná úloha, kdežto za proces pak skupina takových vláken, které k sobě logicky patří

FIM UHK

Vlákna/procesy jádra

- Jejich smyslem je vykonávat kód jádra na úrovni procesu s plným přístupem k paměťovému adresnému prostoru jádra
- Úlohy vlákna jádra:
  - Správa pracovních front
  - Obsluha SW přerušení
  - Automatické načtení modulů
  - Asynchronní operace I/O
  - Zpožděný zápis
  - Odkládání paměťových stránek atd.

FIM UHK

Zdroje

- NEMETH , Evi, SNYDER, Garth, HEIN, Trent R. Linux : kompletní příručka administrátora. 1. vyd. Brno : Computer Press, 2004. 828 s. ISBN 80-7226-919-4.
- Linux : Dokumentační projekt. 4. vyd. Brno : Computer Press, 2007. 1334 s. ISBN 978-80-251-1525-1.
- JELÍNEK, Lukáš. Jádro systému Linux : kompletní průvodce programátora. 1. vyd. Brno : Computer Press, 2008. 686 s. ISBN 978-80-251-2084-2.

Univerzita Hradec Králové
Fakulta informatiky a managementu

Děkuji za pozornost…