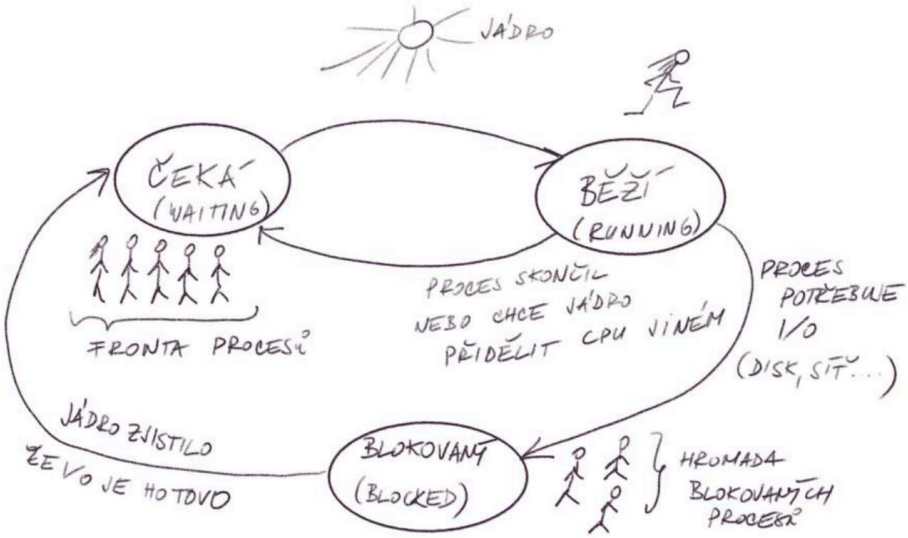
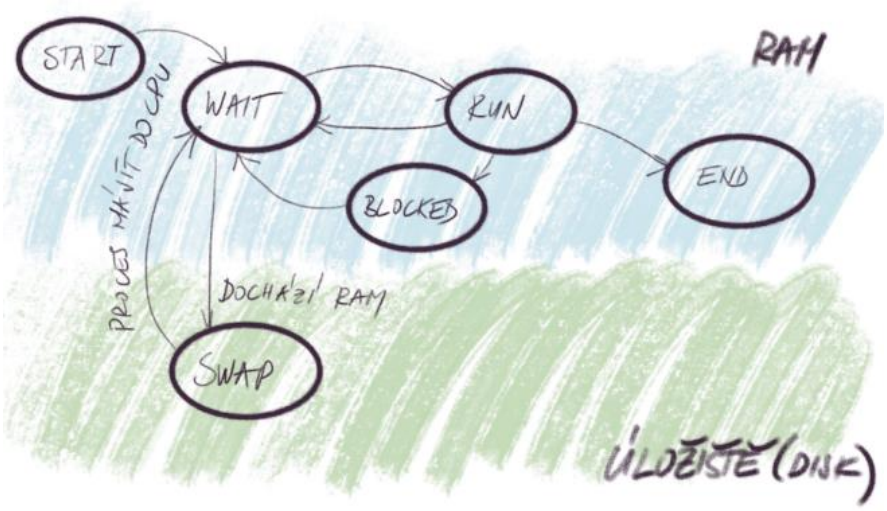
Procesy

Co je proces

Proces je konkrétní spuštěný program, jenž má přidělen prostor v operační paměti. Proces může běžet, být spuštěn, čekat, přistupovat k paměti, prochází různými stádii. V souvislosti s procesem rozlišujeme dva typy dat: data procesu a data o procesu. Data procesu zahrnují kód procesu, hodnoty proměnných, data nebo ukazatele na soubory. Tato data jsou uložena v nechráněné části paměti. Data o procesu jsou informace, které si o procesu vede operační systém. Jedná se například o stav procesu, číslo procesu, nastavení paměti, otevřené soubory, atd. A protože je přepínání procesů úkolem jádra, jsou tyto informace v chráněné části paměti. Každý proces má svoje unikátní číslo (PID). Většina OS přiděluji PID podle toho, kdy byly spuštěny – první spuštěný proces mívá nejnižší PID. Některé systémy ale rozlišují procesy systému a uživatele a vyhradí jim jiný rozsah čísel.

Životní cyklus procesu





Starting: bylo požádáno o spuštění procesu, probíhá kopírování dat z úložiště do paměti, je přiděleno PID.

Ready/wait: proces je ve frontě a čeká na přidělení procesorového času.

Running: procesu byl přidělen procesorový čas a je vykonáván jeho kód.

Blocked: proces požádal o nějakou operaci, která vyžaduje čas (např. přístup k nějakému zařízení).

Swapped: data procesu jsou přesunuta z operační paměti do úložiště. Stává se to v případě, že dochází operační paměť a pro tento proces bylo vyhodnoceno, že je nepravděpodobné, že mu v bude v dohledné době přidělen procesorový čas.

Zombie: proces je vyřazen z fronty cekající na procesor, ale nebude zahájeno mazání jeho dat. Čeká se na další pokyn.

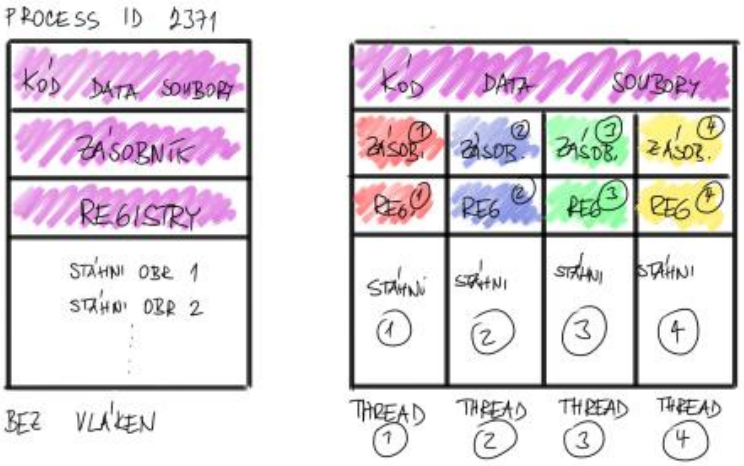
Terminated: Bylo požádáno o ukončení procesu. Odkaz na procesor je odstraněn z fronty procesů čekajících na procesor, ale v paměti jsou stále data procesu, které je potřeba smazat. Proces tedy ještě existuje, ale jíž nebude spuštěn.

Synchronizace procesů

V dokumentu „chytrý jako robot“ je, ať začneme sypat informace o souběhu a uváznutí, což je ale popsáno dole. Tak moc nevím, co sem napsat.

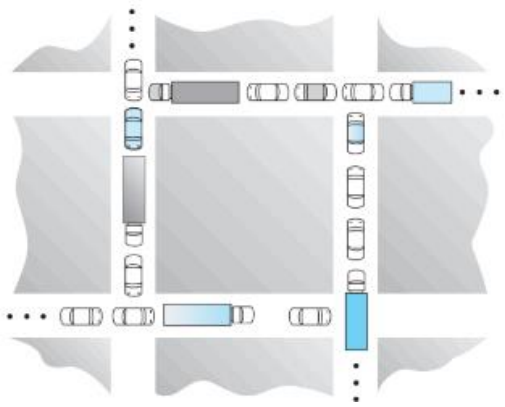
Vlákna

Chceme-li proces dělit na další činnosti, které spolu budou sdílet paměť, využijeme vlákna. Vlákno je tedy něco jako proces, ale na rozdíl od procesů spolu vlákna v rámci jednoho procesu sdílejí přístup k datům procesu. Jediné, co spolu vlákna nesdílejí, jsou své úseky v paměti, kam si vlákna ukládají svůj kontext, dále nesdílejí své zásobníky, protože každé vlákno nezávisle na ostatních realizuje skoky v kódu a jsou i nezávisle přerušována.



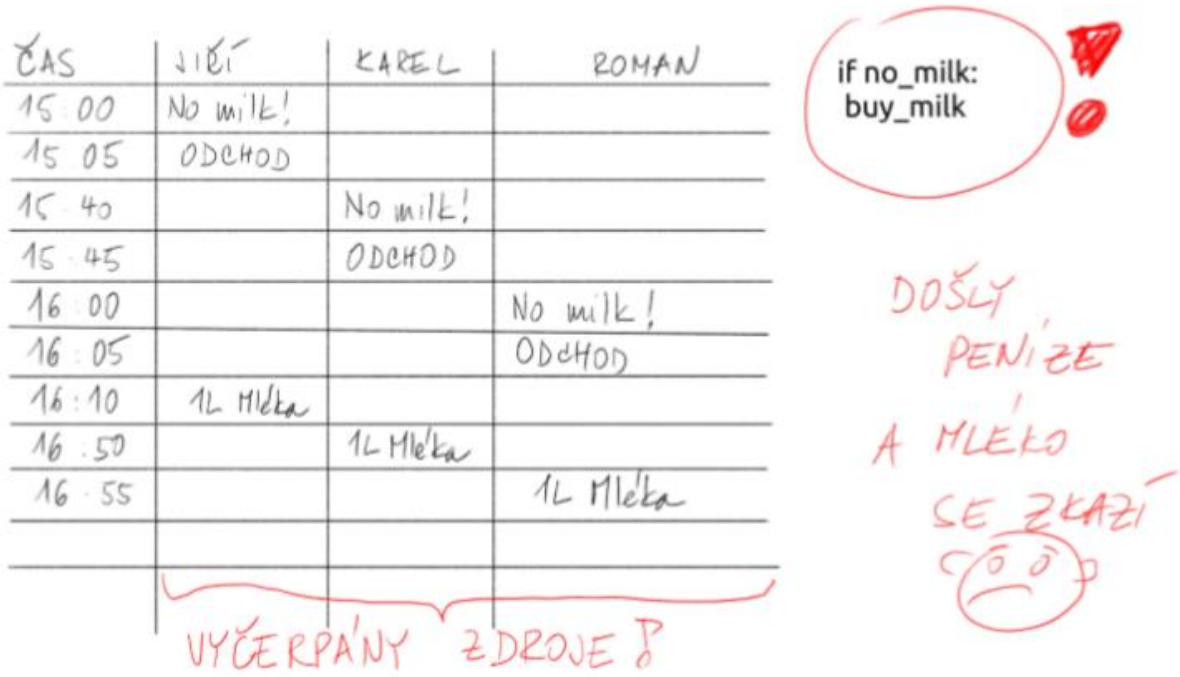
Uváznutí a souběh v OS

Uváznutí: proces 1 potřebuje k pokračování své činnosti zdroj A, jenže ten drží proces 2, který chce zdroj B, který zase drží proces 1. A protože byly procesy naprogramovány tak, aby držely svoje zdroje a čekali na uvolnění dalšího, tak dojde k uváznutí a musí přijít zásah zvenčí.



Jedná se tedy o situaci, kdy se procesy zastaví, protože si navzájem blokují zdroje, bez kterých nemohou pokračovat v práci. Takovéto uváznutí dokáže OS detekovat, pokud ví, jaké prostředky byly komu přiděleny. Pokud uváznutí nastane, máme několik možností, jak postupovat. Můžeme cíleně odebrat jednomu z procesů jeho zdroje. Zde ale přichází velký problém a to ten, komu budou zdroje odebrány. Z toho důvodu se tento přístup nepoužívá. Dále můžeme cíleně ukončit jeden z procesů. V tomto případě je ale možné už tuto práci delegovat na uživatele, který ukončí proces, který souběh zavinil. Procesor tedy strčí hlavu do písku a doufá, že situaci vyřeší uživatel postupným vypínáním procesů. Ač se to zdá absurdní, tak je to podle skript to, co se dnes používá. Avšak mysleme i na zařízení, které nemají uživatelské rozhraní a tím pádem za nás uživatel práci neodvede. Zde přichází na řadu watchdog timer. Princip je poměrně jednoduchý: součástí tohoto řešení je hardwarový čítač, který se s každým taktem sníží o jedna. Po dosažení nuly je systém resetován, protože je čítač napojen na systémový reset. Resetu lze zabránit tak, že nějaký proces čítač resetuje a tím pádem začne „pikat“. Toto mají za úkol hlavně procesy, u kterých je pravděpodobné, že zaviní uváznutí.

Souběh: jedná se o situaci, kdy nějaké dvě entity současně pracují se stejnými daty, aniž by jejich činnost byla jakkoliv synchronizována. Kvůli tomu procesy data náhodně mění a může dojít k poškození dat a ohrožení jejich konzistence.



Jedním z řešení této situace je zámek. V přeneseném slova smyslu lze napasovat zámek i na příklad s mlékem: na lednici je zámek a klíč. Jiří zamkne lednici a klíč si bere s sebou. Ostatní nemohou zjistit stav mléka, tak čekají. Zámek z celého nákupu mléka udělá atomickou operaci, kterou nelze přerušit dalšími dotazy na stav mléka nebo výpravou pro mléko. Nevýhoda ale je, že pokud Jiřího cestou srazí auto, tak už lednici nikdo neotevře. Tím může dojít k uváznutí. Zámky máme dvojího druhu: mutex, což je případ, ve kterém je proces čekající na zámek odložen mezi blokované procesy, nebo proces aktivně čeká ve frontě a sám opakovaně testuje hodnotu zámku. Tomu se říká spin lock.

Dále pak máme semafor. Slouží k přidělování přístupu ke zdroji, jehož je v systému k dispozici určitý počet kusů (třeba síťové spojení). Princip je stejný jako parkoviště. Žádáme tedy o zdroj (parkovací místo) a čekáme, až se zvedne závora. To samozřejmě závisí na tom, jestli je na parkovišti místo (máme zrovna volný „kus“ onoho zdroje?). Pokud není, tak čekáme. Až se místo vytvoří, bude nám přidělen onen zdroj. Až proces provede metodu release (funkce API OS), znamená to, že auto odjelo z parkoviště a tedy že onen zdroj je volný a může být přidělen dalšímu čekajícímu procesu.