

## Watch Dog a Deadlock

Požadavky na spolehlivost software řídicího počítače. Hlavní smyčka. Princip hlídání běhu programu.

Proti čemu chrání? (hloupost, chyba HW, Deadlock) Tedy z jakých důvodů se může zakousnout procesor/program.

Možno vysvětlit na programu. [[dog.zip](#)]

[[http://en.wikipedia.org/wiki/Watchdog\\_timer](http://en.wikipedia.org/wiki/Watchdog_timer)]

[<http://cs.wikipedia.org/wiki/Watchdog>]

---

**Watchdog** (z angličtiny – „hlídací pes“) je počítačová periferie, která [resetuje](#) systém při jeho zaseknutí. K zaseknutí systému může dojít v důsledku chyby v [hardware](#) nebo [software](#) systému. Program (většinou v hlavní smyčce) periodicky signalizuje watchdogu svůj chod. To se může dít např. zápisem servisního impulsu do watchdogu, v případě některých [jednočipových mikropočítačů](#) také provedením speciální [instrukce](#). Pokud systém určitý čas nesignalizuje chod (typicky milisekundy až [sekundy](#)), pak watchdog způsobí reset systému. Záměrem je většinou přivést systém zpět ze zaseknutého stavu k normální funkci.

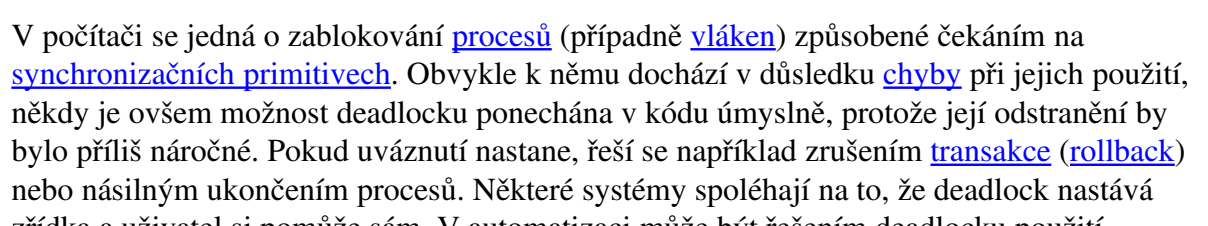
Složitější watchdogy mohou navíc ještě zaznamenávat na nevolatilní (nezávislé na napájení tj. ukládání do energeticky nezávislé paměti) médium ladící informace (např. časy kdy došlo k zresetování systému). Nejčastější použití watchdogů je v [zabudovaných systémech](#), kde jsou mnohdy součástí [mikroprocesoru](#).

Watchdog se rovněž používá pro převedení systému do bezpečného stavu, např. pro vypnutí motorů, elektrických sběrnic s nebezpečným napětím a jiných potenciálně nebezpečných subsystémů. Jednou z možností je, že řídicí jednotka při resetu pozná, že ji zresetoval watchdog, a místo normálního rozběhu pouze uvede systém do bezpečného nouzového stavu.

Watchdog může být realizován pomocí x-bitového [čítače](#) připojeného na [hodinový signál](#) s [frekvencí](#) y MHz. Pokud není čítač přiměřeně často resetován, dojde po uplynutí přetečením

čítače  $\frac{2^x}{y \cdot 10^6}$  sekund k resetu.

## Deadlock

**Deadlock** (česky také **uváznutí**) je odborný výraz pro situaci, kdy úspěšné dokončení nějaké akce je podmíněno předchozím dokončením jiné akce, přičemž tato jiná akce může být dokončena až po dokončení původní akce. Vzniká [paradox](#), často označovaný jako 'Co bylo dříve? Slepice nebo vejce?'.  


V počítači se jedná o zablokování [procesů](#) (případně [vláken](#)) způsobené čekáním na [synchronizačních primitivech](#). Obvykle k němu dochází v důsledku [chyby](#) při jejich použití, někdy je ovšem možnost deadlocku ponechána v kódu úmyslně, protože její odstranění by bylo příliš náročné. Pokud uváznutí nastane, řeší se například zrušením [transakce](#) ([rollback](#)) nebo násilným ukončením procesů. Některé systémy spoléhají na to, že deadlock nastává zřídka a uživatel si pomůže sám. V automatizaci může být řešením deadlocku použití bezpečnostního mechanismu watchdog

Deadlock vzniká při splnění takzvaných čtyř Coffmanových podmínek.  
Tyto podmínky lze řešit následujícím způsobem:

#### Vzájemné vyloučení

Pro mnoho prostředků lze zabránit exkluzivně používání předřazením spoolu, tedy [fronty](#), do které posílají procesy úlohy. Takto je řešeno například používání [tiskárny](#), ale i přístup k [disku](#).

#### Drž a čekej

Proces musí o všechny prostředky, které potřebuje, požádat najednou. Buď je všechny dostane, nebo nedostane ani jeden. Takto postupují [databáze](#) při použití zámků v jazyku [SQL](#) – musí všechny tabulky, které chtějí mít zamčené, zamknout najednou.

#### Neodnímatelnost

Napadení podmínky neodnímatelnosti vede k chaosu. Střídání procesů na procesoru sice na první pohled vypadá jako napadení této podmínky při správě prostředku *čas procesoru*, ale je možné jen díky tomu, že existují okamžiky kdy změnu procesu provést nelze.

#### Čekání do kruhu

Vznik cyklu lze vyloučit například pokud existuje jednoznačné pořadí, v jakém se o prostředky žádá. Na tomto principu je postaveno i hierarchické zamykání, kdy se smí zamykat pouze směrem od kořene stromu dolů.

Pokračování navázat na kritickou sekci a řešení událostí (interrupt)