# Plánování procesů

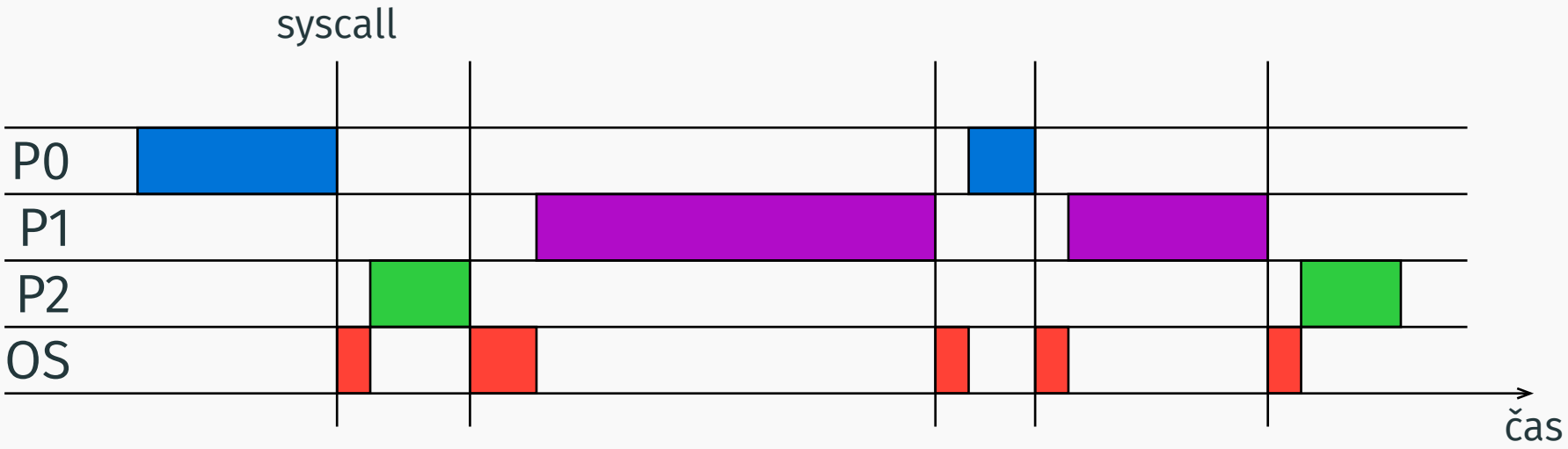
## Metriky, FIFO, SJF, STCF, Round Robin

Milan Radojčić

SSPŠaG – Operační systémy

8. 11. 2025

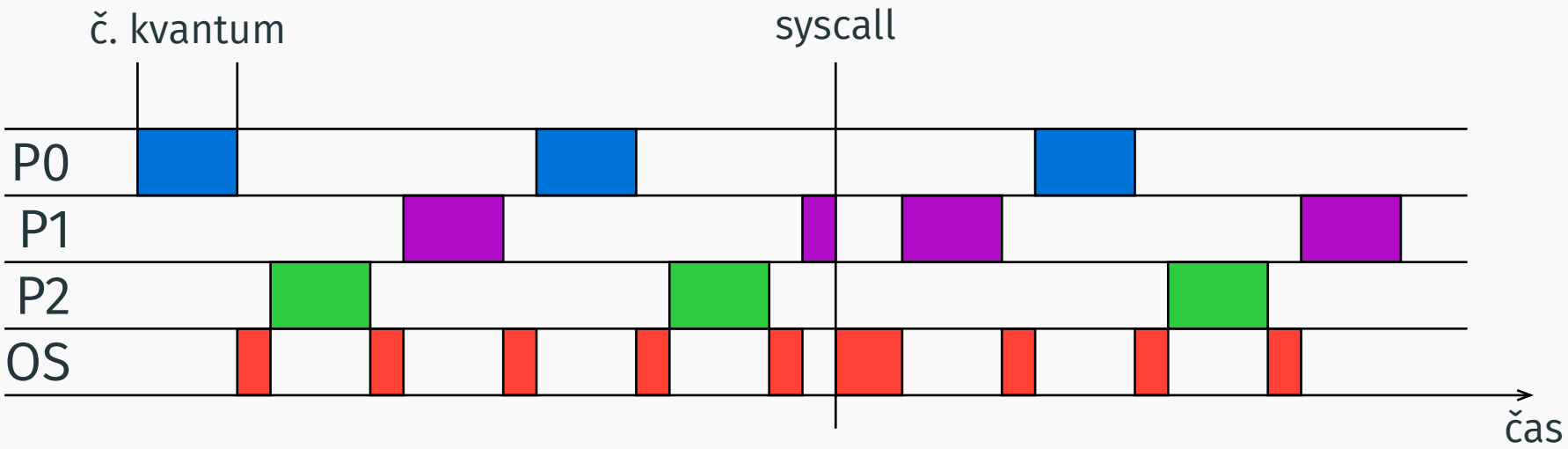
# Kooperativní plánování




Při **kooperativním přístupu** se musí procesy samy vzdát CPU a vrátit kontrolu OS.

Při změně běžícího procesu se musí uložit různé informace aktuálního procesu (např. stav registrů) a obnovit informace nového procesu. Tomu se říká **přepnutí kontextu** (angl. context switch).

# Preemptivní plánování




Při **preemptivním přístupu** je vláknům přiřazeno nějaké **časové kvantum**.[1] Po něm je jim procesor odebrán. Vlákna se mohou vzdát procesoru před vypršením kvanta pomocí systémového volání.

[1] Jak se tohoto přesně docílí v procesoru probereme později.

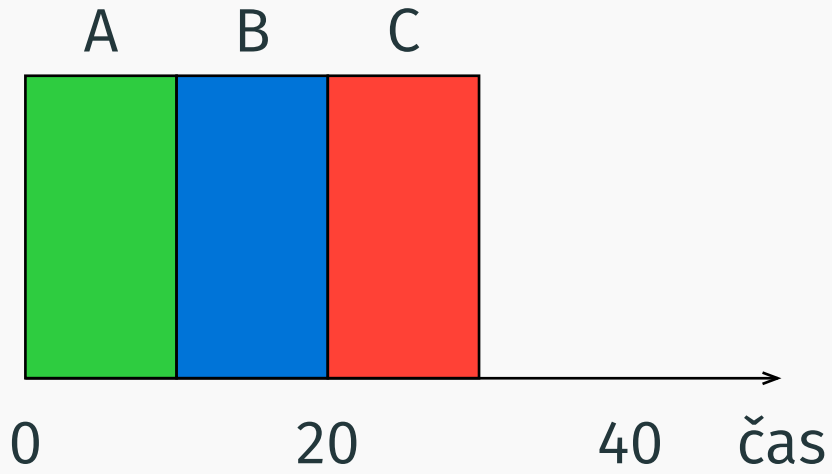
- Teď si ukážeme některé základní plánovací algoritmy.
- Nejprve budeme předpokládat těchto 5 věcí o procesech:
  1. Každý proces běží stejnou dobu.
  2. Každý proces přijde do systému ve stejnou chvíli.
  3. Jakmile se proces spustí, tak běží až než skončí.
  4. Všechny procesy používají jen CPU (neprovádí I/O).
  5. Známe dobu běhu každého procesu.
- Algoritmy budeme taky chtít mezi sebou nějak porovnávat pomocí **metrik**.

**Doba k dokončení (turnaround time)**

$$T_{\text{turnaround}} = T_{\text{completion}} - T_{\text{arrival}}$$

# First In, First Out (FIFO)

Do systému přijdou 3 procesy: A, B a C. Všechny běží stejnou dobu (10s) a přišly do systému ve (skoro) stejnou dobu ($T_{\text{arrival}} = 0$). Řekněme, že A přišel o trochu dřív než B a B přišel o trochu dřív než C. Jak budou procesy naplánovány?



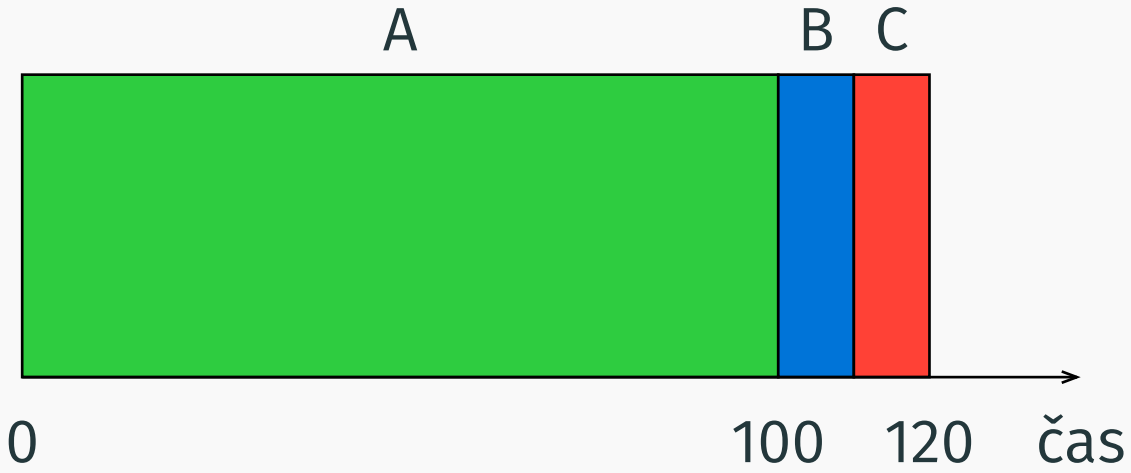

Jak bude vycházet průměrná doba k dokončení?

$$T_{\text{turnaround}}^{\text{average}} = \frac{10 + 20 + 30}{3} = 20\text{s}$$

# Kdy FIFO není optimální

Nyní povolíme, že procesy můžou běžet různou dobu. Napadají vás příklady, kdy průměrná doba k dokončení bude velmi špatná pro FIFO?

Například stejná situace jako na předchozím slidu, ale A bude mnohem delší (např. 100s).




Kolik je pro tento příklad doba k dokončení?

$$T_{\text{turnaround}}^{\text{average}} = \frac{100 + 110 + 120}{3} = 110\text{s}$$

# Shortest Job First (SJF)

Ukažme si, jak by SJF naplánovalo předchozí příklad.


Jak vychází doba k dokončení?

$$T_{\text{turnaround}}^{\text{average}} = \frac{10 + 20 + 120}{3} = 50\text{s}$$

# Kdy SJF není optimální

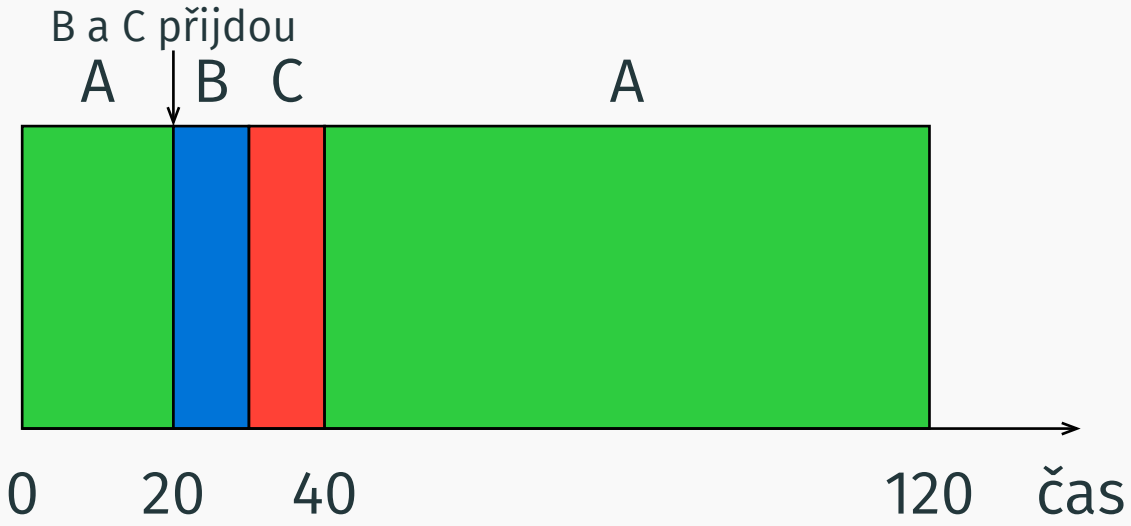
Nyní povolíme, že procesy mohou přijít do systému v různou dobu (ne jen $T_{\text{arrival}} = 0$). V jakém případě nebude SJF optimální?

Např. v situaci, kdy nejprve přijde do systému dlouhý proces a poté dva kratší.


$$T_{\text{turnaround}}^{\text{average}} = \frac{100 + (110 - 20) + (120 - 20)}{3} = 96.66\text{s}$$

# STCF

Abychom mohli vylepšit náš algoritmus, musíme povolit přeplánování procesů (jeden proces pozastavíme a spustíme jiný). To vede na algoritmus **Shortest Time-to-Completion First** (STCF).




$$T^{\text{average}}_{\text{turnaround}} = \frac{120 + (30 - 20) + (40 - 20)}{3} = 50\text{s}$$

# Doba odezvy

- Do teď jsme plánovali jednoduché procesy, které vždy skončí a nejsou interaktivní.
- Co kdybychom začli plánovat **interaktivní procesy**?
  - Co by nás mělo zajímat?

## Doba odezvy (response time)

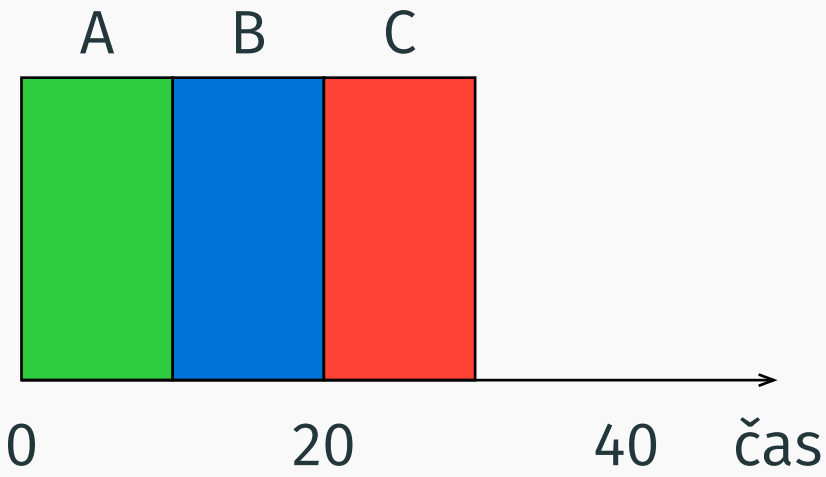
$$T_{\text{response}} = T_{\text{firstrun}} - T_{\text{arrival}}$$

Jak jsou na tom plánovací algoritmy, co jsme si popsali, z hlediska doby odezvy?

# Round Robin

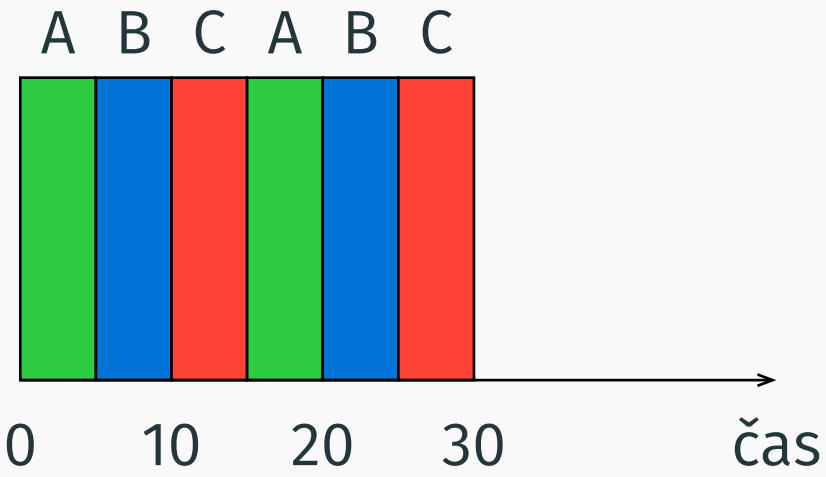
Máme zase 3 procesy, které přišly ve stejnou chvíli a každý trvá 10s.

**FIFO**




$$T^{\text{average}}_{\text{response}} = \frac{10 + 20}{3} = 10\text{s}$$

**Round Robin**




$$T^{\text{average}}_{\text{response}} = \frac{5 + 10}{3} = 5\text{s}$$

- Ale co doba k dokončení?

$$T^{\text{average}}_{\text{turnaround}} = \frac{10 + 20 + 30}{3} = 20\text{s}$$

$$T^{\text{average}}_{\text{turnaround}} = \frac{20 + 25 + 30}{3} = 25\text{s}$$

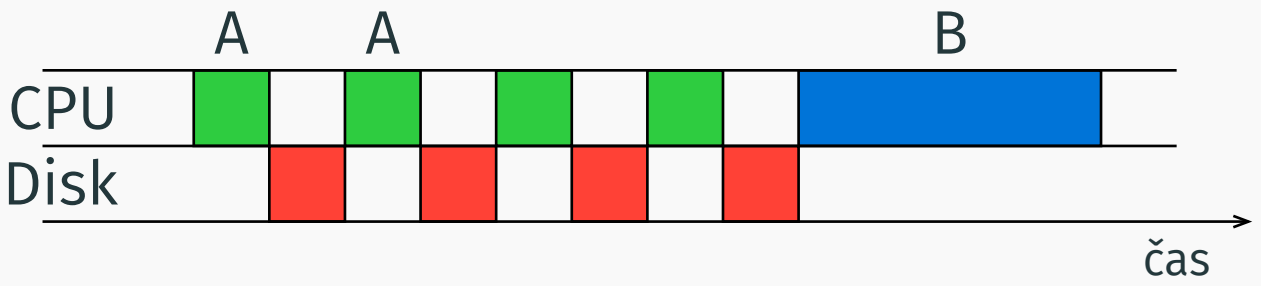
# Doba k dokončení vs doba odezvy

- Vidíme, že tyto algoritmy umí optimalizovat dobu k dokončení nebo dobu odezvy, **ale ne oboje**.
  - Pro programy, které provádí hodně výpočtů (**CPU-bound**) je lepší **doba k dokončení**.
  - Pro programy, které jsou interaktivní (**IO-bound**) je lepší **doba odezvy**.

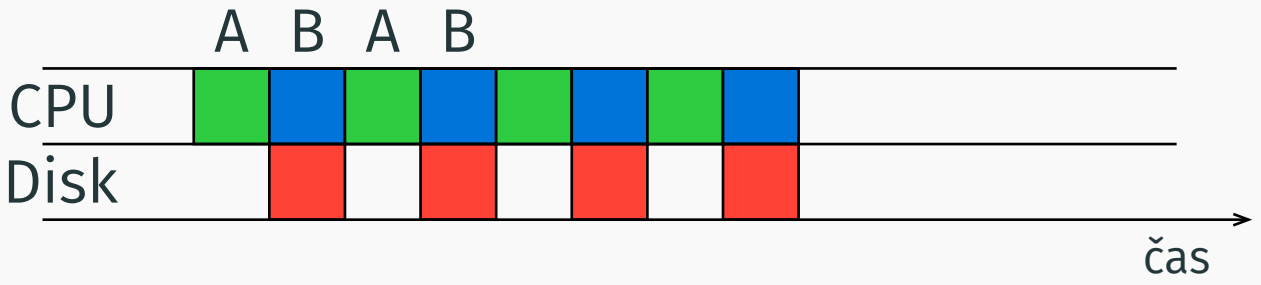
# Jak na IO?

Nyní povolíme procesům provádět IO operace. Jak správně plánovat procesy v takovém případě?

**Špatné využití CPU**




**Správné využití CPU**




Při IO operacích chceme procesy **prokládat**.

# Co dál?

- Pořád nám zbývá zbavit se posledního předpokladu.
  - Doopravdy neznáme dobu běhu každého procesu.
- Jak tohle překonat si povíme příště, to povede na **multi-fevel feedback queue** (MLFQ).
  - To bude náš první opravdový plánovač, který je stále používán[<2>] (např. v Solaris).

[<2>] s nějakými úpravami

# Co si po dnešku pamatovat

- Základní plánovací algoritmy
  - FIFO, Shortest Job First (SJF), Shortest Time-to-Completion First (STCF), Round Robin
- Doba k dokončení a doba odezvy
- **Obecná problematika plánování**
  - **Jak se stavět k interaktivním procesům, jak k CPU-bound?**