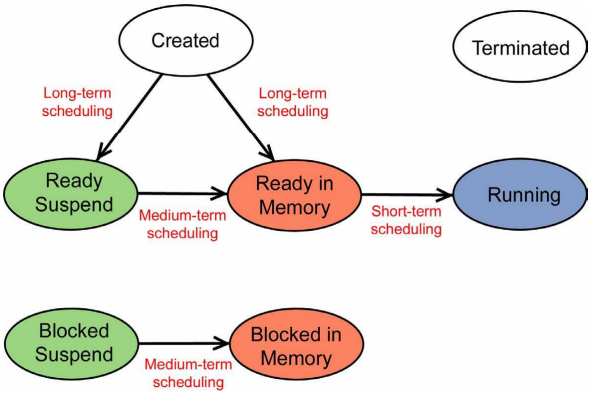
Operační systémy

Přednáška 3: Plánování procesů a vláken

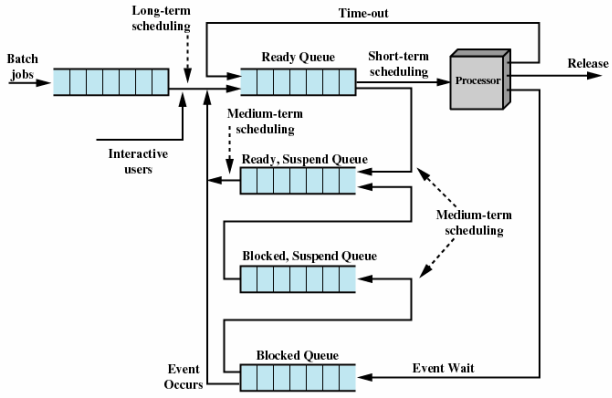
Plánovací algoritmy

- Určují, který z čekajících procesů (vláken) bude pokračovat.
- **Typy plánování**
  - **dlouhodobé (long-term scheduling)**
    - určuje, které programy budou zpracovány systémem
  - **střednědobé (medium-term scheduling)**
    - součást odkládací funkce
  - **krátkodobé (short-term scheduling)**
    - při přerušení od časovače nebo V/V zařízení, systémové volání, signály,..
  - **V/V (I/O scheduling)**
    - určuje, který V/V požadavek bude obsloužen volným V/V zařízením

Typy plánování



Typy plánování (2)



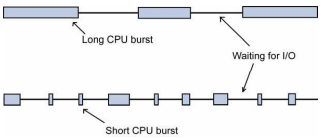
Kritéria krátkodobého plánování

- **Uživatelské hledisko**
  - **doba zpracování (turnaround time)**
    - doba, která uplyne od spuštění do ukončení procesu
  - **doba odpovědi (response time)**
    - doba, která uplyne od okamžiku zadání požadavku do doby první reakce
  - **dosažení meze (deadlines)**
    - zaručení ukončení procesu do dané meze
  - **předvídatelnost (predictability)**
    - proces by měl pokaždé běžet „stejně“ dlouho bez ohledu na zatížení systému

Kritéria krátkodobého plánování (2)

- **Systémové hledisko**
  - **propustnost (throughput)**
    - počet procesů dokončených za jednotku času
  - **vyžití procesoru (processor utilization)**
  - **spravedlivost (fairness)**
    - zamezit hladovění (starvation)
  - **prosazení priorit (enforcing priorities)**
    - možnost uplatnit zvolenou plánovací strategii
  - **vyvážení V/V (balancing resources)**
    - snaha udržet V/V prostředky maximálně vytížené

Typy procesů



- **Procesy orientované na CPU (CPU-bound processes)**
  - CPU využíváno po dlouhou dobu, málo časté čekání na V/V.
- **Procesy orientované na V/V (I/O-bound processes)**
  - CPU využito po krátkou dobu, velmi časté čekání na V/V.
- Klíčový faktor je doba využití CPU
- Procesy mají tendenci stále více využívat V/V prostředky než CPU, ale CPU se zdokonalují rychleji než V/V prostředky.
  - musíme **vytěžovat V/V prostředky co nejvíce**
  - když proces orientovaný na V/V a chce CPU, měl by ho co nejdříve dostat

Kdy plánujeme?

- **Vytvoření nového procesu**
- **Ukončení běžícího procesu**
- **Běžící proces je zablokován z důvodu provádění systémového volání**
  - Důvod zablokování může hrát roli při plánování (např. proces A čeká až proces B opustí kritickou sekci).
  - Většinou ale plánovač nemá k dispozici tuto informaci.
- **Přerušení od V/V zařízení**
  - Vybereme nějaký „ready“ proces nebo „blocked“ proces čekající na V/V?
- **Přerušení od časovače**
  - Plánovací rozhodnutí při každém přerušení od časovače nebo při *k*-tém přerušení od časovače.

Strategie plánování

- **Plánování s předbíháním (preemptive scheduling)**
  - Běžící proces je zablokován automaticky po uplynutí časového kvanta.
  - Jediná možná strategie v RT-systémech a více uživatelských systémech.
- **Plánování bez předbíhání (nonpreemtive scheduling)**
  - Proces běží dokud nepožádá o nějakou službu jádro nebo neskončí.
  - Proces může blokovat systém a musí „spolupracovat“ pouze když žádá službu jádra.
  - Používalo se v dávkových systémech.

Plánování v dávkových systémech

- **First-Come First-Served (FCFS)**
  - Jednoduché plánování bez předbíhání pomocí jedné FIFO fronty procesů.
  - Když je běžící proces zablokován, první proces ve frontě bude pokračovat.
  - Když se zablokovaný proces stane opět „ready“, je zařazen na konec fronty.
  - **Výhody:** jednoduché na pochopení i implementování.
  - **Nevýhody:** FCFS může zpomalit procesy orientované na V/V.

Plánování v dávkových systémech (2)

- **Shortest Job First (SJF)**
  - Plánování bez předbíhání, které předpokládá, že **doba výpočtu je známa předem**.
  - Plánovač spouští nejdříve procesy s nejmenší dobou výpočtu.
  - SJF minimalizuje průměrnou dobu zpracování.

Proces	A	B	C	D
Čas výpočtu [min]	8	4	4	4
Pořadí zpracování A,B,C,D				
Doba zpracování	8	12	16	20
Průměrná doba zpracování 14 min				
Pořadí zpracování D,C,B,A				
Doba zpracování	20	12	8	4
Průměrná doba zpracování 11 min				

Plánování v dávkových systémech (3)

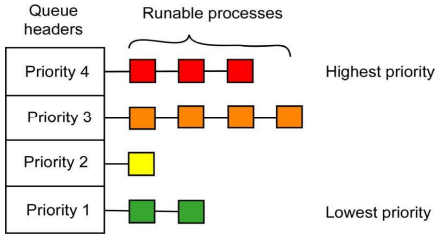
- **Shortest Remaining Time Next (SRT)**
  - Verze SJF pro plánování s předbíháním.
  - Plánovač přidělí CPU procesu s nejmenším časem výpočtu.
  - Když přijde požadavek na spuštění nového procesu, plánovač porovná časy výpočtu nového procesu a aktuálního procesu.
  - Proces s menším časem výpočtu získá CPU.
  - **Nevýhoda:** riziko hladovění procesů s velkým časem výpočtu.

Plánování v interaktivních systémech

- **Round-Robin Scheduling (RR)**
  - Jednoduché plánování s předbíháním.
  - „Ready“ procesy čekají ve frontě FIFO.
  - Každý proces dostane přiděleno CPU na **časové kvantum**.
  - Po uplynutí časového kvanta, běžící proces je pozastaven a uložen na konec FIFO fronty, první proces ve FIFO frontě dostane přiděleno CPU.
  - Délka časového kvanta je důležitý parametr:
    - Krátké čas. kvantum ⇒ nízká efektivita CPU.
    - Dlouhé čas. kvantum ⇒ špatná doba odezvy.
    - Rozumný kompromis je 20-50ms.

Plánování v interaktivních systémech (2)

- **Priority Scheduling (PS)**
  - Každý proces má přidělenu **prioritu**.
  - „Ready“ procesy se stejnou prioritou jsou seskupeny do **prioritních tříd**.
  - **CPU je přiřazováno procesům z nejvyšší prioritní třídy**.
  - Procesy z nejvyšší prioritní třídy se střídají **metodou round-robin**.



Plánování v interaktivních systémech (3)

- **Varianty prioritního plánování**
  - **bez předbíhání**
  - **s předbíháním**
    - **fixní časové kvantum** pro všechny prioritní třídy
    - **různě velká časová kvanta** pro různé prioritní třídy
- **Problém**
  - **hladovění (starvation)**
  - Proces s nízkou prioritou nedostane CPU.
- **Řešení**
  - procesu, který čeká déle než je stanovený limit, je dočasně zvýšena priorita

Příklad: zvýhodnění procesů orientovaných na V/V

- Procesy orientované na V/V stráví většinu času čekáním na dokončení V/V operací.
- Procesy orientované na V/V by měly dostat C/PU co nejdříve, aby mohly opět zahájit V/V operaci.
- **Každému procesu nastavíme dynamickou prioritu $P=t_q/t_u$**
  - t_q je časové kvantum a
  - t_u je čas skutečně strávený na CPU.
- Např. proces, který využil 2ms ze 100ms čas. kvanta dostane prioritu 50. Zatím co proces, který běžel 50ms (ze 100ms čas. kvanta) dostane prioritu 2.

Příklad: multiple queues (MQ)

- Operační systém CTSS
  - **přepínání kontextu bylo velmi pomalé**, neboť počítač (IBM 7094) mohl mít v paměti pouze jeden proces.
  - Proto bylo vhodné nastavit velké časové kvantum, abychom **redukovali režii na přepínání kontextu**.
- **Proces v nejvyšší třídě běží po dobu jednoho čas. kvanta. Proces v následující třídě poběží po dobu dvou čas. kvant, …**
- **Pokud proces využije celé čas. kvantum, bude přesunut do následující nižší třídy.**
- Např. proces, jehož doba výpočtu je 100q, dostane 1q, 2q, 4q, 8q, 16q, 32q, 64q. Z posledního čas. intervalu (64q) využije pouze 37q.
  - v MQ dojde pouze k 7 přepnutím kontextu
  - v RR by došlo ke 100 přepnutím kontext.

Plánování v interaktivních systémech (4)

- **Shortest Process Next (SPN)**
  - Modifikace SJF pro interaktivní OS.
  - Čas výpočtu procesu se odhaduje na základě minulého chování procesu.
  - Proces s nejmenším odhadem času výpočtu dostane CPU.
  - Odhad:

$$T_0 \ldots\ldots\ldots\ldots \text{odhad},$$
$$T_1 \ldots\ldots\ldots\ldots \text{čas následujícího behu}$$
$$kT_0 + (1-k)T_1 \ldots \text{nový odhad } (0 \le k \le 1).$$

  - Pro k=1/2, dostaneme

$$T_0,\quad T_0/2 + T_1/2,\quad T_0/4 + T_1/4 + T_2/2,\quad T_0/8 + T_1/8 + T_2/4 + T_3/2$$

  - Využívá se techniky **stárnutí (aging).**

Plánování v interaktivních systémech (5)

- **Fair-share scheduling (FSS)**
  - vyšší numerická hodnota priority znamená nižší prioritu
  - v systému existuje 1,…,n skupin
  - každá skupina k, 1≤k≤ n,
    - má přiřazenu váhu W_k, kde $W_1+W_2+\ldots+W_k=1$
    - v časovém intervalu i používala CPU po dobu $GCPU_k(i)$
  - každý proces j
    - patří právě do jedné skupiny k
    - má nastavenu fixní base prioritu B_j
    - v časovém intervalu i používal CPU po dobu $CPU_j(i)$
    - během časového intervalu i má prioritu $P_j(i)$
  - na začátku každého čas. intervalu i se přepočtou hodnoty a naplánuje se další proces podle priority $P(i)$

$$CPU_j(i) = \frac{CPU_j(i-1)}{2}$$
$$GCPU_k(i) = \frac{CPU_k(i-1)}{2}$$
$$P_j(i) = B_j + \frac{CPU_j(i)}{2} + \frac{GCPU_k(i)}{4 \times W_k}$$

Unix - plánování procesů

- prioritní plánování s prioritními třídami
- vyšší numerická hodnota priority znamená nižší prioritu
- procesy dostávají přiděleny „různě“ velká čas. kvanta
- priorita každého procesu se přepočítává každou sekundu
  - B_j umožňuje definovat prioritní úrovně
  - $nice_j$ uživatelem modifikovatelná část priority

$$CPU_j(i) = \frac{CPU_j(i-1)}{2}$$
$$P_j(i) = B_j + \frac{CPU_j(i)}{2} + nice_j$$

- Příklad:
  - procesy A, B a C jsou spuštěny ve stejném okamžiku (B_j=0 a $nice_j$=0)
  - každý proces běží 1 sekundu
  - časovač přeruší systém 60 krát za sekundu a inkrementuje čítač běžícího procesu

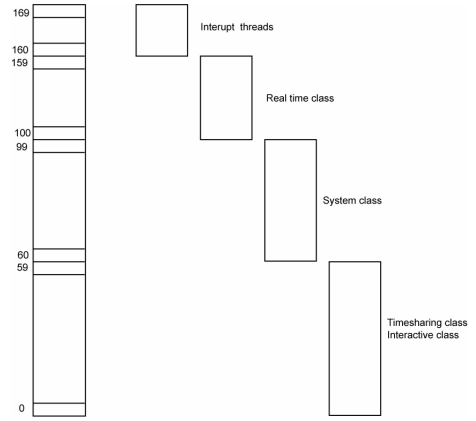
Unix – plánování procesů(2)

Time	Process A Priority	Process A CPU Count	Process B Priority	Process B CPU Count	Process C Priority	Process C CPU Count
0	60	0 1 2 . . 60	60	0	60	0
1	75	30	60	0 1 2 . . 60	60	0
2	67	15	75	30	60	0 1 2 . . 60
3	63	7 8 9 . . 67	67	15	75	30
4	76	33	63	7 8 9 . . 67	67	15
5	68	16	76	33	63	7

Solaris – plánování procesů

- **Plánovací třídy**:
  - **Timesharing (TS)**
    - normální uživ. procesy,
    - prioritní plánování s předbíháním,
    - procesy orientované na CPU mají prioritu nižší než procesy orientované na V/V
    - procesy, které nedostaly CPU během určitého času, budou mít zvýšenou prioritu.
  - **Interactive (IA)**
    - podobné jako TS, ale priorita procesů v aktivním okně se rychleji zvyšuje.
  - **System (SYS)**
    - systémové vlákna,
    - prioritní plánování bez předbíhání,
    - fixní priorita.
  - **Real Time (RT)**
    - RT procesy,
    - nejvyšší priorita (pouze obsluha přerušení má vyšší)
    - plánování s předbíháním,
    - fixní priorita.

Solaris – plánování procesů (2)



Solaris – plánování procesů (3)

- Informace o parametrech plánování

dispadmin -c TS -g

# ts_quantum	ts_tqexp	ts_slpret	ts_maxwait	ts_lwait	PRIORITY LEVEL
500	0	10	5	10	# 0
500	0	11	5	11	# 1
500	1	12	5	12	# 2
500	1	13	5	13	# 3
500	2	14	5	14	# 4
500	2	15	5	15	# 5
450	3	16	5	16	# 6
450	3	17	5	17	# 7
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
50	48	59	5	59	# 58
50	49	59	5	59	# 59

Solaris – plánování procesů (4)

- **ts_quantum**
  - Aktuální časové kvantum pro danou prioritu.
- **ts_tqexp**
  - Nová priorita pro proces, který využil celé čas. kvantum.
- **ts_slpret**
  - Nové priorita pro proces, který nevyužil celé čas. kvantum.
- **ts_maxwait**
  - Pokud proces nedostal během tohoto intervalu CPU, tak bude mít novou prioritu **ts_lwait.**
- **PRIORITY LEVEL**
  - Aktuální priorita procesu.

Solaris – plánování procesů (5)

- Jak změnit plánovací třídu procesu?

  dispadmin

- Jak změnit prioritu procesu?

  priocntl -s -c třída -t časové_kvantum -p priorita program

  nice +10 příkaz

MS Windows XP – plánování procesů

- Prioritní plánování s předbíháním
- Různě velká časová kvanta (normálně 2 x clock interval)

- **Úrovně priorit (priority level)**
  - **kernel**
    - 0 – system level (zero page thread)
    - 1-15 – dynamic levels
    - 16-31 – real-time levels
  - **API**
    - process base priority class (Idle, Below Normal, Normal, Above Normal, High, Real-time)
    - relative thread priority (Time-critical, Highest, Above-normal, Below-normal, Lowest, Idle)

- Proces je běžně spouštěn s base priority (24,13,10,8,6, nebo 4)

MS Windows XP – plánování procesů (2)