06.
Čtenáři – písaři
Plánování procesů

ZOS 2006

Problém čtenářů a písařů

- modeluje přístup do databáze
- rezervační systém (místenky, letenky)

- množina procesů, souběžné čtení a zápis
  - souběžné čtení lze
  - výhradní zápis (žádný další čtenář ani písař)

```
var
   m=1 : semaphore;    {mutex}
   w=1: semaphore;     {přístup pro zápis }
   rc = 0: integer;    { počet čtenářů }

procedure writer;
begin
   P(w);
      // zapisuj
   V(w)
end;
```

```
procedure reader;
begin
    P(m);
    rc := rc + 1;
    if rc = 1 then P(w);
    V(m);
        // čti
    P(m);
    rc := rc - 1;
    if rc=0 then V(w);
    V(m)
end;
```

Čtenáři – písaři popis

- čtenáři
  - první čtenář provede P(w)
  - další zvětšují čítač rc
  - po “přečtení” čtenáři zmenšují rc
  - poslední čtenář provede V(w)
- semafor w
  - zabrání vstupu písaře, jsou-li čtenáři .. přednost písařů
  - zabrání vstupu čtenářům při běhu písaře
    - prvnímu zabrání P(w)
    - ostatním brání P(m)
- toto řešení je s předností čtenářů
  - písaři musí čekat, až všichni čtenáři skončí

Implementace zámků v operačních a databázových systémech

- přístup procesu k souboru nebo záznamu databázi

- výhradní zámek (pro zápis)
  - nikdo další nesmí přistupovat
- sdílený zámek (pro čtení)
  - mohou o něj žádat další procesy

- granularita zamykání
  - celý soubor x část souboru
  - tabulka x řádka v tabulce

Implementace zámků v OS

Linux, UNIX lze zamknout část souboru funkcí

fcntl (fd, F_SETLK, struct flock)

```
int fd; struct flock fl;
fd = open("testfile", O_RDWR);
fl.l_type = F_WRLCK;                    - zámek pro zápis
fl.l_whence = SEEK_SET;                 - pozice od začátku souboru
fl.l_start = 100; fl,l_len = 10;        - pozice, kolik
fcntl (fd,F_SETLK, &fl);          - zamkneme pro zápis
 // vrací -1 pokud se nepovede
```

Implementace zámků v OS

odemknutí

fl.l_type = F_UNLCK;	- odemknutí
fl.l_whence = SEEK_SET;	- pozice od začátku souboru
fl.l_start = 100; fl,l_len = 10;	- pozice, kolik
fcntl (fd,F_SETLK, &fl);	- zamkneme pro zápis

operace

F_SETLK	- set / clear lock
F_GETLK	- info o zámku
F_SETLKW	- nastavení zámku, čeká

Zámky v DB systémech

např. s každým záznamem databáze sdružen zámek

funkce:

db_lock_r(x)	zámek pro čtení
db_lock_w(x)	uzamkne záznam x pro zápis
db_unlock_r(x)	odemčení záznamu x
db_unlock_w(x)	dtto

čtenáři – písaři s předností písařů

```
type zamek = record
wc, rc: integer := 0;              // počet písařů a čtenářů
mutw: semaphore := 1;              // chrání přístup k čítači wc
mutr: semaphore := 1;              // chrání přístup k čítači rc
wsem: semaphore := 1;              // blokování písařů
rsem: semaphore := 1;
                                   // je-li písař, blokuje 1. čtenáře
rdel: semaphore := 1;
                                   // blokování ostatních čtenářů
end;
```

```
procedure db_lock_w(var x: zamek);

    // uzamčení záznamu pro zápis
begin
    P(x.mutw);
    x.wc:=x.wc+1;
    if x.wc=1 then P(x.rsem);  -- 1.písař zablokuje 1. čtenáře
    V(x.mutw);
    P(x.wsem);                 --  blokování písařů
end;
```

```
procedure db_unlock_w(var x: zamek);
  // odemčení zápisů pro zápis

begin
    V(x.wsem);                    -- odblokování písařů
    P(x.mutw);
    x.wc:=x.wc-1;
    if x.wc=0 then V(x.rsem); -- poslední písař pustí 1.čten.
    V(x.mutw)
end;
```

```
procedure db_lock_r(var x: zamek);
begin
    P(x.rdel);                          -- nejsou blokováni ostatní čtenáři
    P(x.rsem);                          -- není blokován 1. čtenář
    P(x.mutr);
    x.rc:=x.rc+1;
    if x.rc=1 then P(x.wsem);           -- 1. čtenář zablokuje písaře
    V(x.mutr);
    V(x.rsem);
    V(x.rdel)
end;

procedure db_unlock_r(var x: zamek);
begin
    P(x.mutr);
    x.rc:=x.rc-1;
    if x.rc=0 then V(x.wsem);           -- odblokuje písaře
    V(x.mutr)
end;
```

Další problémy meziprocesové komunikace

- problém spícího holiče (přednáčka č.4)
- problém populárního pekaře (Lamport 1974)
- plánovač hlavičky disku
- další probrané
  - problém hladových filozofů
  - producent – konzument
  - čtenáři - písaři

Plánování procesů

Základní stavy procesu

- běžící
- připraven – čeká na CPU
- blokován – čeká na zdroj nebo zprávu

- nový (new) – proces byl právě vytvořen
- ukončený (terminated) – proces byl ukončen

Správce procesů – udržuje tabulku procesů

Záznam o konkrétním procesu – PCB (Process Control Block) – souhrn dat potřebných k řízení procesů

Plánování procesů

- plánovač vs. dispatcher
- dispatcher předává řízení procesu vybranému short time plánovačem
  - přepnutí kontextu
  - přepnutí do user modu
  - skok na vhodnou instrukci daného programu
- více připravených procesů k běhu – plánovač vybere, který spustí jako první
- plánovač procesů (scheduler), používá plánovací algoritmus (scheduling algorithm)

Plánování procesů - vývoj

- dávkové systémy
  - spustit další úlohu, nechat ji běžet do konce
- systémy se sdílením času
  - procesy běžící na pozadí
  - interaktivní procesy
- kombinace obou systémů (dávky, interaktivní procesy)

- přednost interaktivních procesů
  - odesílání pošty x zavírání okna

Střídání CPU a I/O aktivit procesu

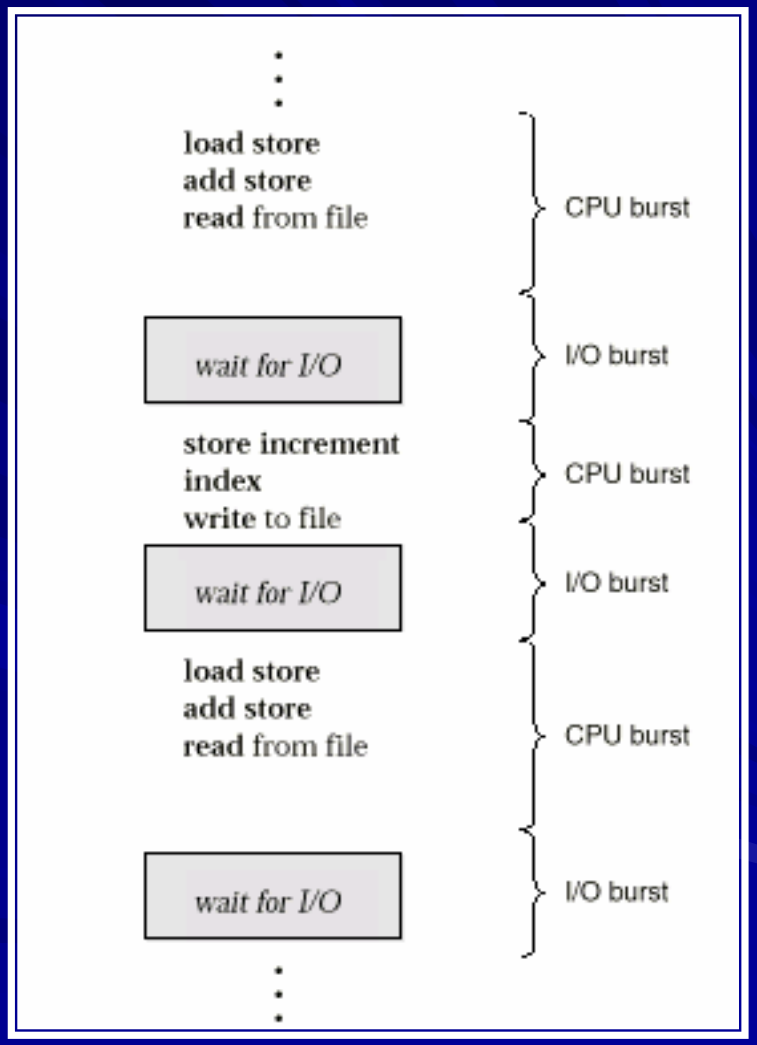
Během vykonávání procesu

- CPU burst (vykonávání)
- I/O burst (waiting)
- střídání těchto fází
- končí CPU burstem

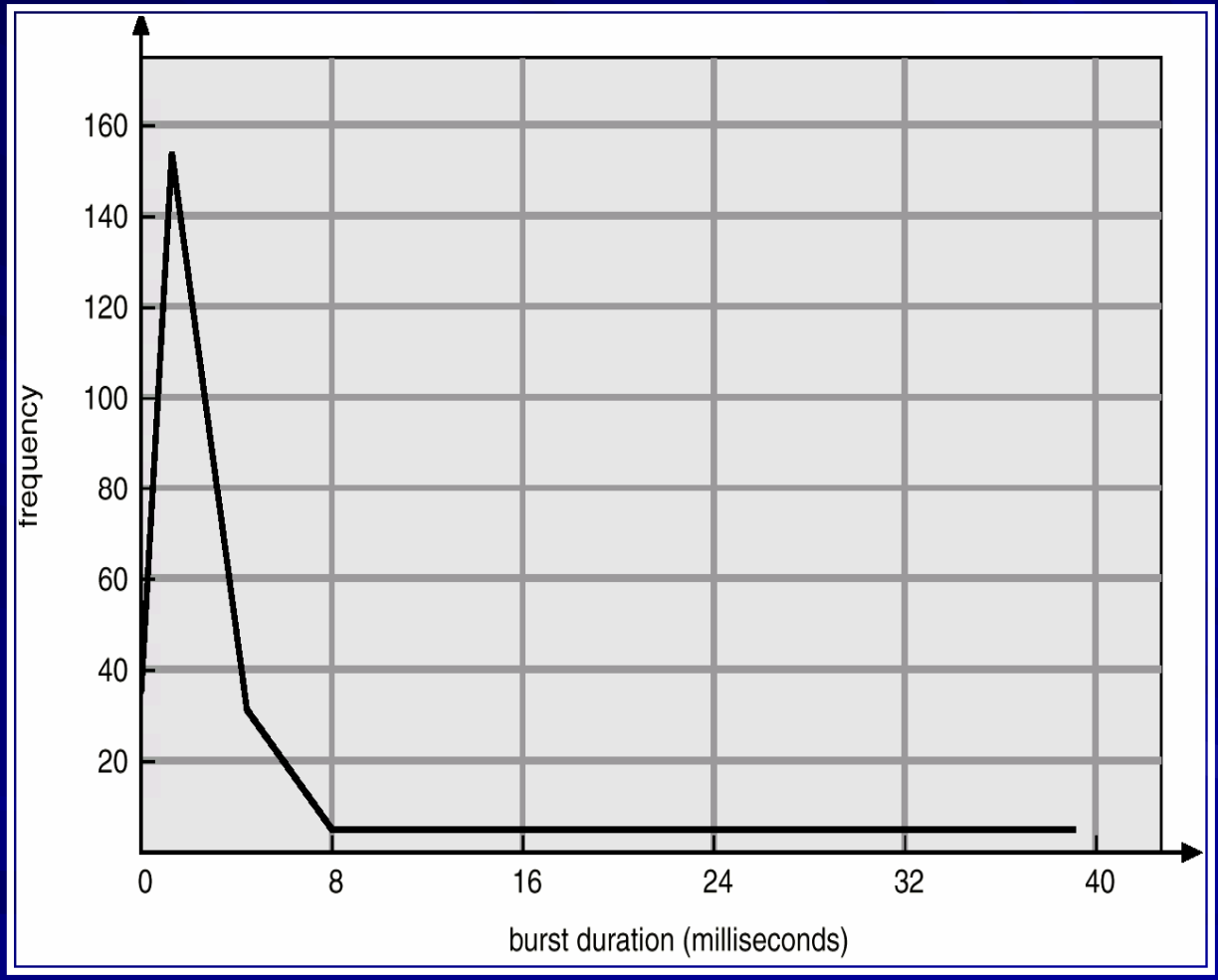
typicky

hodně krátkých burstů

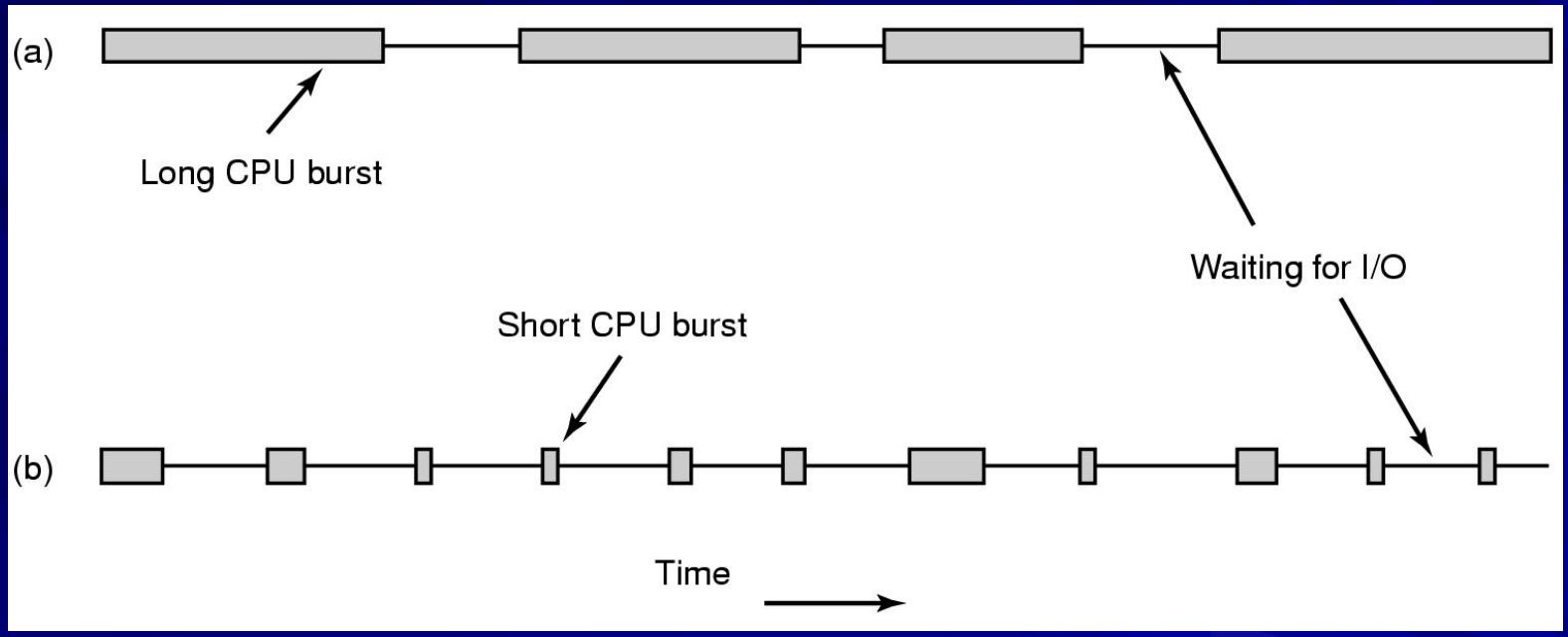
málo dlouhých



Histogram CPU burstů



Plánování



- a CPU-bound process
- an I/O bound process

Preemptivní vs. non-preemptive plánování

- Non-preemptivní
  - každý proces dokončí svůj CPU burst
  - proces si podrží kontrolu nad CPU, dokud se jí nevzdá (I/O čekání, ukončení)
  - lze v dávkových systémech, není příliš vhodné pro time sharingové
  - Win 3.x non-preemptivní (kooperativní) plánování
  - od Win95 preemptivní
  - od Mac OS 8 pro PowerPC – preemptivní
  - na některých plaformách stále – preempce vyžaduje spec. hw, např. timer

Preemptivní vs. non-preemptive plánování

- Preemptivní plánování
  - proces lze přerušit během CPU burstu a naplánovat jiný
  - dražší implementace kvůli přepínání procesů (režie)

Otázky preemptivní plánování

- koordinace přístupu ke sdíleným datům
- preempce jádra OS
  - přeplánování ve chvíli, kdy se manipuluje s daty (I/O fronty) používanými jinými funkcemi jádra..
  - UNIX
    - čekání na dokončení systémového volání
    - nebo na I/O
  - výhodou jednoduchost jádra
  - nevýhodou výkon v RT a multiprocessingu

Cíle plánování

All systems

- Fairness - giving each process a fair share of the CPU
- Policy enforcement - seeing that stated policy is carried out
- Balance - keeping all parts of the system busy

Batch systems

- Throughput - maximize jobs per hour
- Turnaround time - minimize time between submission and termination
- CPU utilization - keep the CPU busy all the time

Interactive systems

- Response time - respond to requests quickly
- Proportionality - meet users' expectations

Real-time systems

- Meeting deadlines - avoid losing data
- Predictability - avoid quality degradation in multimedia systems

Zajímavosti

- V roce 1973 shut-down systému IBM 7094 na MITu
- našli low priority proces, založený ….. a dosud nespuštěný

Zajímavosti

- ..v roce 1967 ..

Pokračování

- viz material p6plan.pdf