Meziprocesová komunikace, roury, sdílená paměť, semafory, zasílaní zpráv - implementace.

Thursday, May 30, 2013 8:28 AM

Řada aplikací se skládá z mnoha spolupracujících procesů. Ty mezi sebou komunikuje a sdílejí informace.

Jádro OS musí poskytovat mechanismy, které to umožní – nazýváme je *prostředky meziprocesové komunikace*. Jejich **účelem je**:

- Přenos údajů
- Sdílení dat
- Oznámení vzniku událostí
- Sdílení prostředků
- Sledování a sdílení běhu procesu, např. při ladění programu

1. **Signály**
   - Umožňují oznámení procesům o asynchronní události
   - Viz otázka *„Signály“*
2. **Roury (pipes)**
   - Zápis dat na konec roury, čtení dat od začátku

     a. Nepojmenované roury

   - Vytvoření systémovým voláním **`pipe()`** – vrací dva deskriptory, jeden pro čtení, druhý pro zápis
   - Při vytváření procesů jsou deskriptory roury děděné
   - do roury může zapisovat i číst z ní více procesů, data jsou čtena v pořadí, v jakém byla zapsána
   - Procesy mohou **komunikovat** prostřednictvím roury, **pokud byla vytvořena společným předchůdcem**
   - Po skončení všech předchůdců přestává roura existovat

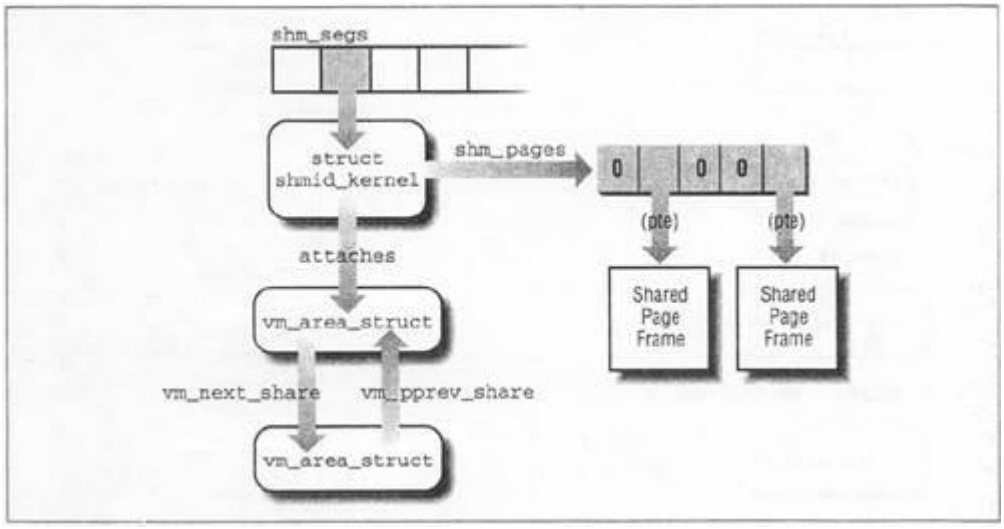
     b. Pojmenované roury, FIFO roury
   - **Perzistentní,** existují jako soubory, i když je nepoužívají žádné procesy
   - FIFO musí být explicitně zrušen, jako obyč soubory – **`unlink`**
   - Oproti obyč souborům jsou přečtená data odstraněna a z pohledu komunikace mají stejnou sémantiku jako nepojmenované roury
   - Vytvoření FIFO souboru: **`mknode(cesta, mód, zařízení)`**, **`mkfifo(cesta, mód)`**
     - Mód = obvyklá oprávnění
     - Zařízení = pro vytváření speciálních souborů pro zařízení
   - FIFO jsou pak otevřena sys voláním **`open()`** – vrací deskriptor souboru
   - do FIFO jde zapisovat sys voláním **`write()`** nebo z něj číst sys voláním **`read()`**

Následující způsoby mají podobnou implementaci – jsou identifikovány IPC (inter-process comunication) klíčem (obdoba cesty k souboru) a zpřístupňovány identifikátory IPC (obdoba deskriptoru souboru) – ty nejsou vázány na proces a nemění se v průběhu života objektu; získání identifikátoru IPC voláním `shmget(), semget(), msgget()`

3. **Sdílená paměť**
   - Oblast paměti, která je sdílená více procesy
   - Sdílená data jsou v paměti fyzicky uložena jen jednou, procesy je mají namapované do vlastního virtuálního paměťového prostoru (např. Používají DLL)
   - Proces ji **vytvoří/získá** voláním **`shmid = shmget(klíč, velikost, příznak);`**
     - Shared memory id, shared memory get
   - Proces **připojí oblast** na virtuální adresu voláním:
     `adr = shmat(shmid, shmadr, shmpříznak);`
     - „shared memory at“
     - `shmadr` je návrh adresy pro připojení oblasti; pokud je `shmadr` nula, jádro adresu

vybere
- skutečná adresa je návratová hodnota

- **Odpojení** oblasti:
  - `shmdt(shmaddr);`



4. **Semafory** - synchronizační primitivum s celočíselným čítačem a metodami P() a V()
   - **`semid = semget(klíč, počet, příznak)`**
     - vrátí identifikátor pole semaforů o velikost počet
   - **`stav = semop(semid, sops, nsops)`**
     - atomicky vykoná operace nad polem semaforů
     - `sops` je ukazatel na pole s `nsops` prvky typu `sembuf`

```
struct sembuf {
    unsigned short sem_num;
    short sem_op;
    short sem_flg;
};
```

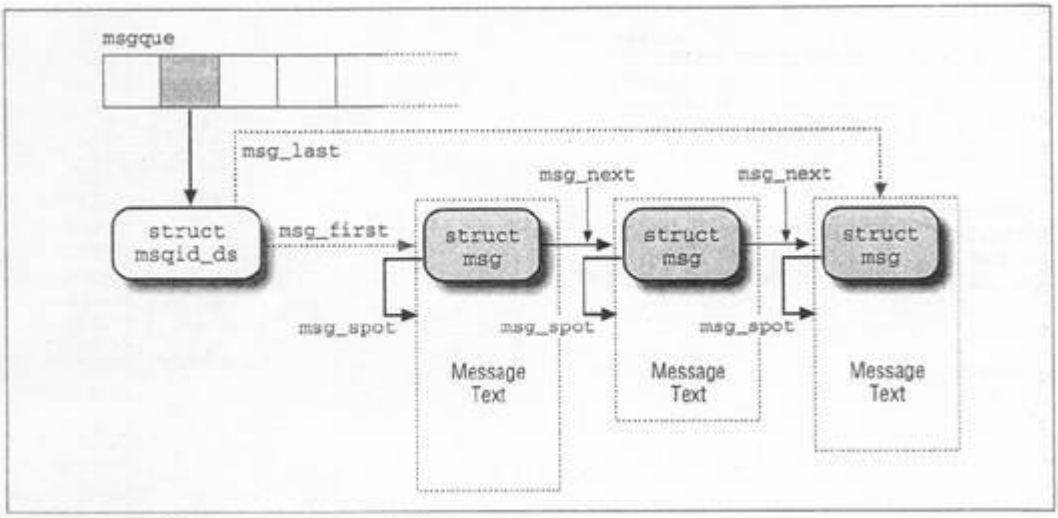
5. **Zasílání zpráv**
   - Procesy komunikují prostřednictvím zpráv
   - Zpráva vytvořená procesem je zaslána do fronty zpráv, je tam, dokud ji jiný proces nepřečte
   - Zpráva obsahuje 32bitový typ a data zprávy
   - Typ zprávy umožňuje selektivní výběr zpráv z fronty
   - Synchronní/asynchronní, blokující/neblokující
   - Proces **získá/vytvoří zprávu** voláním:
     - **`msgqid = msgget(klíč, příznak);`**
     - „message queue id“
   - Zpráva se **uloží do fronty** voláním:
     - **`msgsnd(msgqid, msgp, počet, příznak)`**
     - msgp = pointer na zprávu obsahující typ zprávy, který je následován
     - počet = velikost zprávy včetně typu v bytech
   - zprávy jsou ve frontě v pořadí jejich příchodu
   - **výběr zpráv** z fronty voláním:
     - **`počet = msgrcv(msgqid, msgp, maxpct, msgtyp, příznak)`**
     - je-li čtená zpráva delší než maxpct, je oseknutá
     - msgtyp = 0 → vrátí se první zpráva z fronty
     - msgtyp kladný → vrátí první zprávu typu msgtyp
     - msgtyp záporný → vrátí se první zpráva nejnižšího typu než je abs hodnota `msgtyp`

   - Na Windows:

     - SendMessage(HWND okna, int message, WPARAM. LPARAM) - blokující, čeká na vyzvednutí zprávy

     - SendNotifyMessage/PostMessage - neblokující

- Každé okno má svou smyčku zpráv while(GetMessage(..)) - blokující
- PeekMessage - neblokující



IPC message queue data structures

From <https://d.docs.live.net/e3534876709763a3/Dokumenty/ZCU/Statnice/Statnice.docx>