

Semestrální práce z X360SY

Michal Turek

19. Překladač (implementace pomocí procesů)

V systému je n překladačů. Klienti vstupují do systému a alokují si překladač. Pokud není žádný překladač volný, pak klient odchází ze systému.

Pokud má klient svého překladače, pak generuje náhodné znaky a-z a A-Z a zapisuje je do fronty. Překladač vybírá znaky z fronty a transformuje malé znaky na velké a opačně. Pokud překladač přeloží všechny znaky pak se zablokuje dokud nepřijde další klient.

Výstup: Informace o překladačích, klientech, frontách a přeložených znacích.

Struktura kódu

Aplikace je naprogramována objektově v jazyce C++. Kód je rozdělen souborů:

```
[woq@woq ~/skola/x360sy/sem_processes]$ ls -l *.h *.cpp
all.h
csemaphore.cpp
csemaphore.h
csharedmemory.cpp
csharedmemory.h
ctranslater.cpp
ctranslater.h
main.cpp
```

Jména by měla být samovysvětlující. Třída klienta nebyla použita, protože by obsahovala pouze jednu metodu a komunikace s třídou překladače by byla o dost složitější.

Uživatelské nastavení

Aplikace je neinteraktivní, veškeré nastavení lze provést pomocí konstant umístěných nahoře v souboru main.cpp, které specifikují zapnutí debug módu (vkládá do kódu čekací smyčky, aby se projevila interakce mezi procesy), ID sdílených prostředků, počet procesů překladačů a klientů a maximální počet znaků který může jeden klient odeslat.

```
#define DEBUG

#define SHM_KEY 987654
#define SEM_KEY 589641

const int NUM_TRANSLATERS = 3;
const int NUM_CLIENTS = 30;
const int MAX_CHARS = 100;
```

Sdílené prostředky

Kvůli zobecnění a hlavně zjednodušení kódu byla funkce pro práci se sdílenou pamětí a semaforey zapouzdřeny do tříd, které mají výrazně intuitivnější interface a jsou mnohem snadněji použitelné.

```
class CSemaphore
{
public:
    CSemaphore(key_t key, int num);
    ~CSemaphore();

    void Lock(int which);
    void UnLock(int which);
    void Sleep(int which);
    void Wake(int which);

private:
    int m_id;
    int m_num;
};

class CSharedMemory
{
public:
    CSharedMemory(key_t key, int size);
    ~CSharedMemory();

    char* Attach();
```

```

        void Detach();
        int GetSize() const { return m_size; }

private:
        int m_id;
        int m_size;
        char* m_data;
};

```

Funkcionalita programu

Většina funkcionality programu je soustředěna do třídy překladače, která především zapouzdřuje veškerou práci se sdílenou pamětí a semaforey.

```

class CTranslator
{
public:
    CTranslator(int id, key_t sem_key, key_t mem_key, int mem_size);
    ~CTranslater();

    bool TryToOccupy();// true = byl volny a muze se zacit pouzivat

    void Push(char c);
    char Pop();// Return \0 if empty

    void TranslatorFunc();
    void WaitForJob();
    void NewJob();

private:
    int m_id;// Vypisy

    CSemaphore m_sem;
    CSharedMemory m_shmem;
    int m_queue_size;

    // Pointers to shared memory
    // | m_occupied | m_write_pos | m_read_pos | m_queue |

    bool* m_occupied;
    int* m_write_pos;
    int* m_read_pos;
    char* m_queue;
};

```

Použitá literatura

<http://www.root.cz/serialy/programovani-pod-linuxem-pro-vsechny/>
<http://www.advancedlinuxprogramming.com/>
<http://www.google.com/>