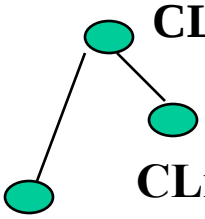
Výukový simulační systém v C++

implementace: moduly SIMSET a SIMULATION

funkce modulu SIMSET: operace pro práci se seznamy

(dle služeb systémové třídy SIMSET jazyka Simula 67)

hierarchie vnitřních tříd:



CLinkage…obecné vlastnosti každého prvku seznamu (ukazatelé na předchůdce a následníka)

CLink…….operace nad prvky seznamu

CHead…..operace nad seznamem

nejdůležitější metody třídy CLink:

CLink* Suc () ..vrací ptr na následující prvek seznamu nebo NULL

CLink* Pred ()..vrací ptr na předcházející prvek seznamu nebo NULL

void Out ()…je-li prvek v nějakém seznamu, je vyjmut, jinak nic

void Follow (CLinkage* X)..zařadí this do seznamu za prvek X (je-li)

void Precede (CLinkage* X)..zařadi this do seznamu před X (je-li)

void Into (CHead* F)..zařadí prvek this na konec seznamu F

nejdůležitější metody třídy CHead:

void Clear () …….. vyjme z daného seznamu všechny prvky

int Cardinal ()……vrací počet prvků v daném seznamu

bool Empty ()…….vrací true, je-li seznam prázdný; jinak false

CLink* First ()……..vrací ptr na první prvek daného seznamu, je-li seznam prázdný, vrací NULL

CLink* Last ()……...vrací ptr na poslední prvek daného seznamu, je-li prázdný, vrací NULL

Funkce modulu Simulation

• služby kvaziparalelního prostředí (dle systémové třídy Simulation jazyka Simula 67)

• základní synchronizační princip: seznam událostí SQS setříděný dle hodnot modelového času

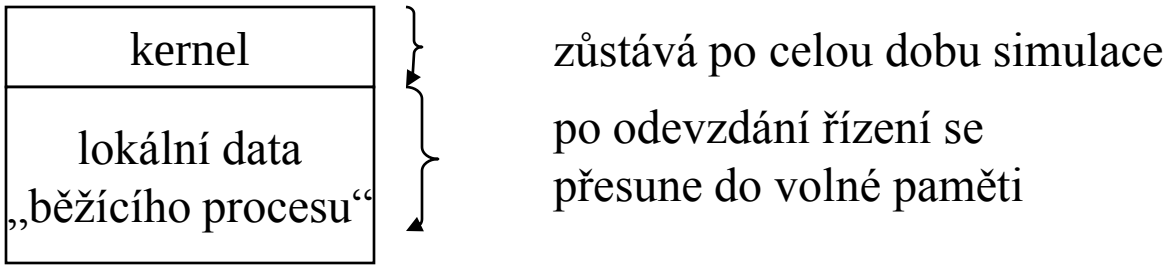
•neexistence operační části třídy : nahrazena virtuální metodou Run ( )

•neexistence hlav. procesu: nahrazena metodou Run () objektu CSimulation

•neexistence koprogramů v C++ : nahrazena vytvořením kontextu za účelem přerušení metody Run () a možností návratu do místa přerušení

1) verse pod OS DOS:

–struktura zásobníku simulačního programu:



–inicializace systému: funkce main () vytvoří prázdný seznam SQS, vygeneruje objekt třídy TSimulation, zařadí jej do SQS a předá řízení funkci kernel

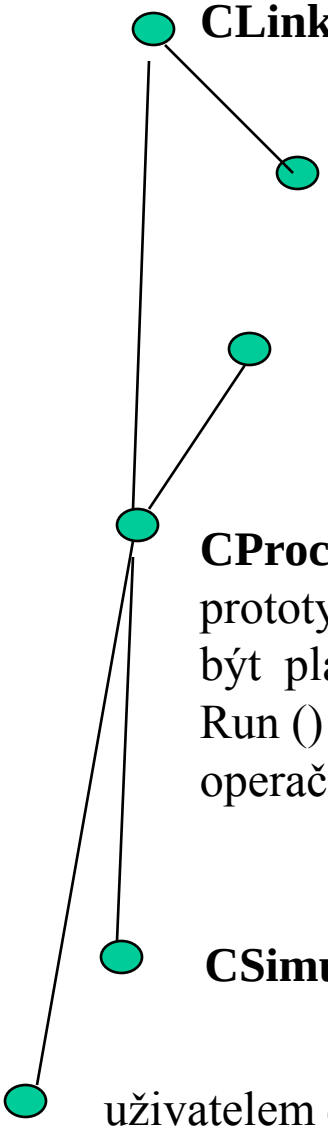
–funkce kernel: najde záznam 1. procesu v SQS, pokud daný proces nebyl dosud spuštěn pak volá metodu Run (), jinak obnoví v zásobníku kontext procesu a předá mu řízení

–„běžící proces“ :před ukončením kterékoliv fáze uloží registry a přesune lokální data ze zásobníku do volné paměti a předá řízení funkci kernel

Výukový simulační systém v C++

2) verse pod OS Windows NT:

- použití vláken s vlastním zásobníkem
- při vygenerování každého procesu se vytvoří vlákno voláním funkce CreateThread () ve stavu SUSPENDED
- hierarchie vnitřních tříd:



CLink (viz modul SIMSET)

CEventnotice……...definuje objekty sloužící jako záznamy událostí v seznamu událostí

CThread….operace s vlákny využívající služeb WINAPI pro vytvoření vlákna, jeho pozastavení a znovuspuštění (navíc vůči versi pod DOS)

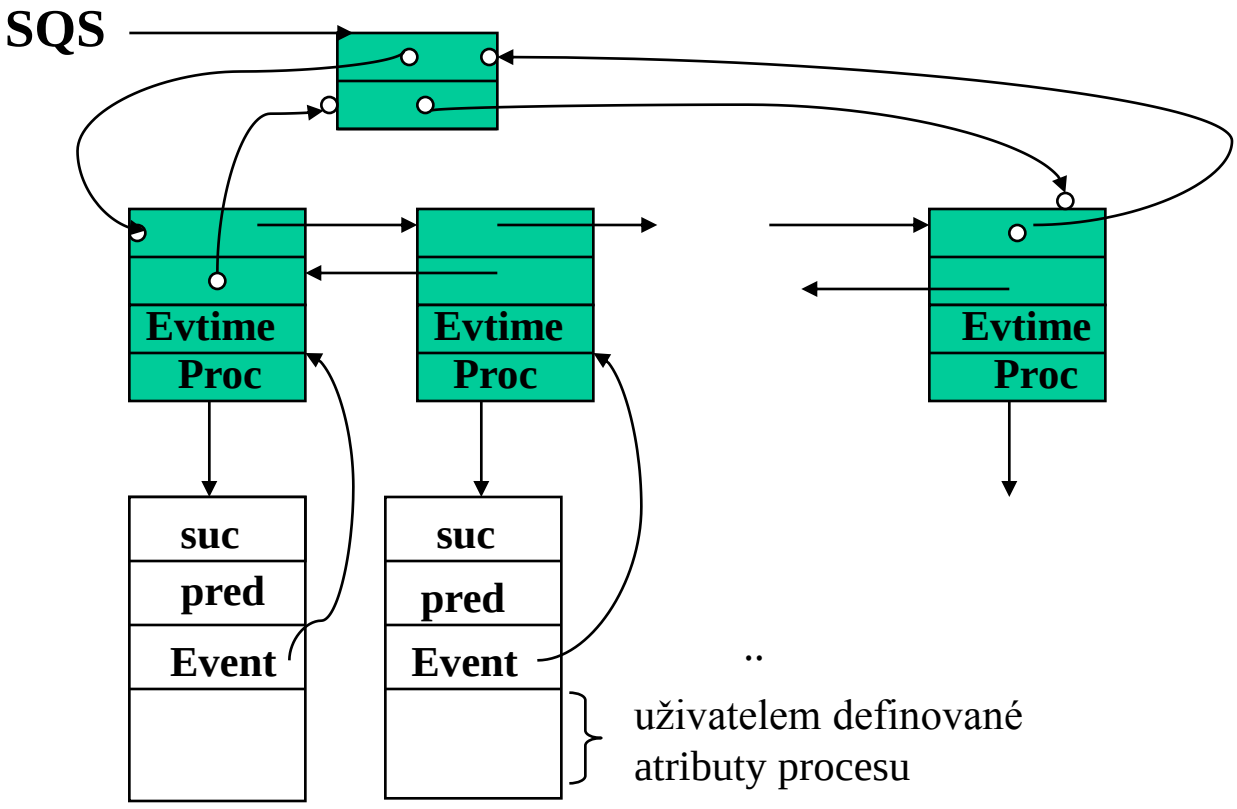
CProcess……...definuje základní vlastnosti pro prototypy uživatelových procesů (objekty, které mohou být plánovány v seznamu událostí) - virtuální metody Run () těchto objektů definuje uživatel (analogie operačních částí tříd „process“ ze Simuly 67)

CSimulation…analogie hlavního programu

uživatelem definované procesy (podtřídy třídy CProcess)

Struktura seznamu událostí

- verse pro Windows:



stavy procesů**:**

aktivní - jde o 1. proces v seznamu událostí

naplánovaný (potlačený) - není aktivní, ale má záznam v SQS

pasivní - proces nemá záznam v seznamu událostí

ukončený - nemá záznam v seznamu událostí a proběhly všechny fáze

testování stavů:

stav	test procesu X
aktivní	X == Current ( )
potlačený	!(X->Idle ( )) && X != Current ( )
pasivní	X->Idle ( ) && !(X-> Terminated ())
ukončený	X -> Terminated ( )

Hlavní vlastnosti třídy CProcess

```
class CProcess : public CLink, public CThread
{private:   void ResumeCurrent ();    // umožní spuštění 1. procesu dle
                                      // SQS a suspenduje právě běžící proces
 public:
 static CHead * SQS;                  // ukazatel na hlavu seznamu událostí
 static CProcess* Currrent ();        // vrací odkaz na aktivní proces
 static CEventNotice* FirstEv();  //vrací odkaz na záznam 1. události v SQS
 static  TIME Time();                 // vrací okamžitou hodnotu modelového času
 CProcess* NextEv ();        //vrací ukazatel na následující proces v SQS
 bool Idle();                 // vrací true, pokud proces nemá záznam v SQS
 bool Terminated ();    //vrací true pokud je proces ukončen
TIME EvTime ();       // vrací čas naplánování (pokud proces není Idle)
 void _activate (……); // dle hodnot parametrů plánuje proces do SQS
 void Hold ( TIME T ); // převede aktivní proces do stavu potlačený -
       // naplánuje jeho příští fázi bez priority a volá ResumeCurrent ()
 void Passivate ();    // převede aktivní proces do stavu pasivní (zruší
     příslušný záznam v SQS) a volá  funkci ResumeCurrent ()
 void Wait (CHead* S);    // převede aktivní proces do stavu pasivní -
              // zařadí jej do seznamu S a provede funkci jako Passivate ()
 void Cancel ( CProcess* p);  //převede proces p do stavu pasivní -
// pokud jde o aktivní proces, pak je funkce stejná jako v případě
     Passivate (), pokud jde o potlačený proces, pak jde o pouhé vyjmutí
     příslušného záznamu ze SQS
virtual void Run ();                        // uživatelův popis procesu
};
```

Implementace funkce Hold (TIME t)

```
void CProcess :: Hold (TIME t)
{
  if  (this !=  Current () ) return;   // lze aplikovat pouze na běžící
                                       // proces
  if  (t <= 0)  return ;               // nelze vrátit v čase
  CEventNotice * e = FirstEv ();  // pointer na 1. zaznam v SQS
  e -> evTime + = t;                   // nastaveni noveho casu
  if ( e -> Suc ()  != NULL )   // existuje vice zaznamu
    { if  (((CEventNotice*) e ->Suc()) ->evTime <= e -> evTime )
            {
             e -> Out ();              // vyjmuti zaznamu z SQS
             e -> Rank (false);     // nove zarazeni zaznamu dle t
            }
    }
    else  ResumeCurrrent () );  // suspenduje bezici vlakno a
                                // aktivuje 1. vlakno z SQS
}
```

Plánování procesů

vkládání procesů do seznamu událostí:

pomocí funkce

_activate (bool reac, CODE code, TIME t, CProcess* r, bool prior)

reac …….odlišuje aktivátory Activate (plánují pouze pasivní procesy) a aktivátory Reactivate (plánují pasivní i potlačené procesy)

pozn.: ukončený proces nelze naplánovat

code……rozlišuje plánovací doložky: přímé plánování (DIRECT), dle model. času (AT, DELAY), na určité místo (BEFORE, AFTER)

prior……plánuje s prioritou, resp. bez priority : tj.před, resp. za procesy které jsou již naplánovány pro stejnou hodnotu modelového času

r……..definuje již existující proces v SQS před, resp. za který se bude plánovat pomocí doložky BEFORE, resp. AFTER

- makra pro aktivátory Activate:

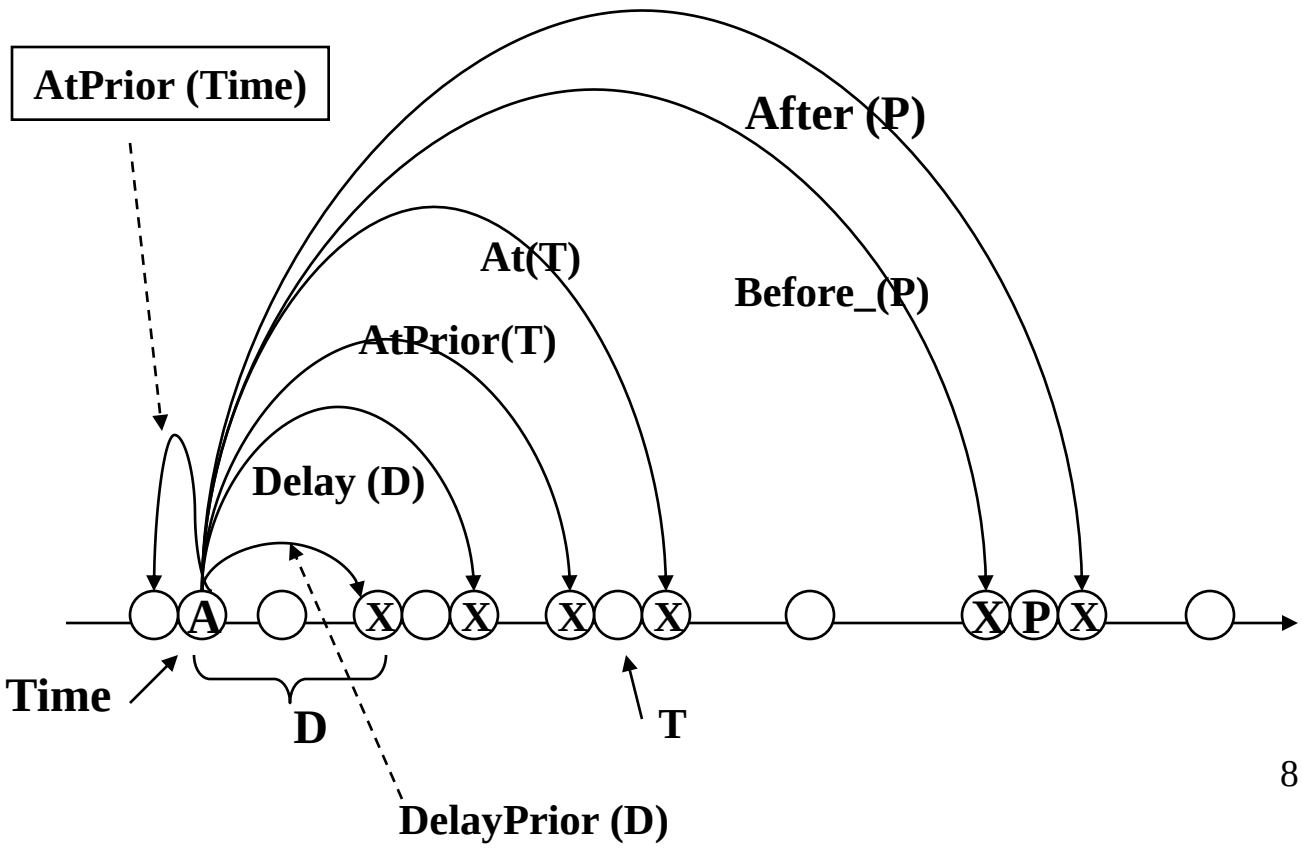
```
#define Activate () _          _activate (false, DIRECT, 0, NULL, false )
#define ActivateAt (t)         _activate (false, AT, t , NULL, false )
#define ActivateAtPrior (t)    _activate (false, AT, t, NULL, true )
#define ActivateDelay (t)      _activate (false, DELAY, t, NULL, false )
#define ActivateDelayPrior (t)
                               _activate (false, DELAY, t, NULL, true )
#define ActivateBefore (r)     _activate (false, BEFORE, 0, r, false )
#define ActivateAfter (r)      _activate (false, AFTER, 0, r, false )
```

Plánování procesů

-

makra pro aktivátory Activate:

```
#define Reactivate ()            _activate (true, DIRECT, 0, NULL, false )
#define ReactivateAt (t)         _activate (true, AT, t, NULL, false )
#define ReactivateAtPrior (t)    _activate (true, AT, t, NULL, true )
#define ReactivateDelay (t)      _activate (true, DELAY, t, NULL, false )
#define ReactivateDelayPrior (t)
                                 _activate (true, DELAY, t, NULL, true )
#define ReactivateBefore (r) _activate (true, BEFORE, 0, r, false )
#define ReactivateAfter (r)      _activate (true, AFTER, 0, r, false )
```



Plánování procesů

poznámka: **bezprostřední plánování** - vede na přímé předání řízení

X-> Activate ();

X-> ActivateAtPrior (Time());

X -> ActivateDelayPrior (0);

X -> ActivateBefore (Current());

stejný efekt

příklad:

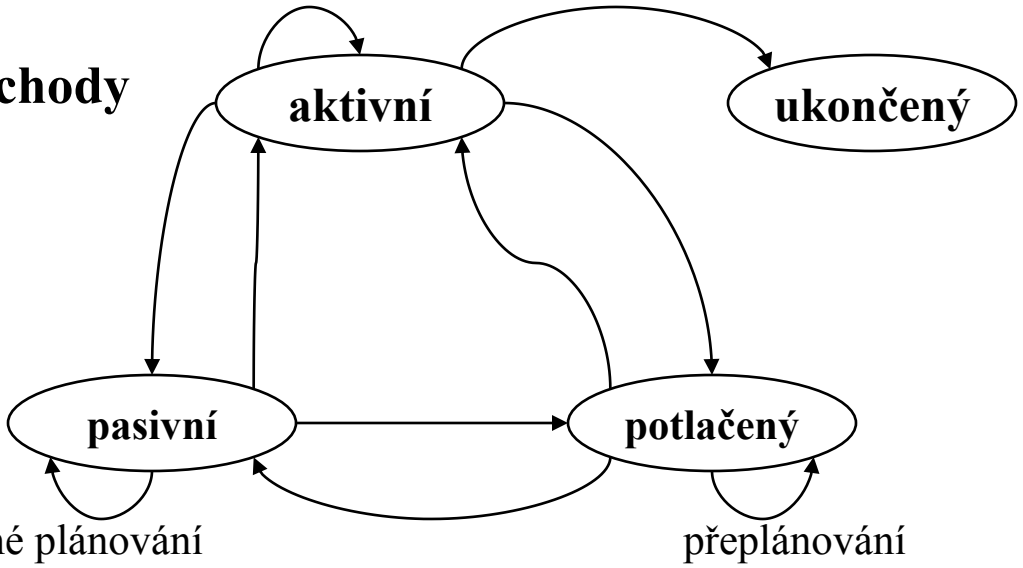
X -> ReactivateDelay (T);

a) X odkazuje na aktivní proces => analogie Hold (T)

b) X odkazuje na proces, který není aktivní:

- pasivní => nové naplánování: jako X->ActivateDelay (T)
- potlačený => dojde k přeplánování (tj. vyjmutí ze SU a k novému naplánování); jde o efekt jako Cancel (X); X->ActivateDelay (T);

možné přechody mezi stavy procesů:



Transformace náhodných čísel

generování hodnot náhod. veličin s požadovaným rozložením:

1) generování rovnoměrně rozložených celých čísel $x_i \in \langle 0, 2^\beta - 1 \rangle$
   (pomocí kongruenčních metod)
2) transformace celých čísel x_i na reálná čísla u_i z intervalu $\langle 0,1)$
   ($u_i = x_i / 2^\beta$), $u_i \in U(0, 1)$
3) transformace reálných čísel $u_i \in U(0, 1)$ na požadované rozložení:

- **funkce pro některá ze standardních rozložení:**

int draw (float p)...nula - jedničkové rozložení: P[1]=p, P[0]=1-p

int randint (int a, int b)….rovnoměrné na intervalu a,b

float uniform (int a, int b)…rovnoměrné na intervalu a,b

float normal (float s, r)…normální rozložení:s:stř.hod., r:rozptyl

float negexp (float s)…..exponenciální rozložení, stř. hodn.:1/s

int poisson (float s)…..poissonovo rozložení, stř. hodn.: s

- **funkce pro experimentálně zjištěná rozložení:**

int discrete (float a [ ])

  – generuje diskrétní hodnoty 1,2,3,…. dle schodovité distribuční funkce jejíž funkční hodnoty jsou definované parametrem typu pole: F(x) = a (x)

float linear (float a [ ], float b [ ])

  – generuje hodnoty spojité náhodné veličiny dle po úsecích lineární distribuční funkce definované pomocí dvou parametrů typu pole: F (b(x)) = a (x)