# Тема 13. Сервисы и серверные приложения

Под системой типа **«клиент-сервер»** подразумевают систему централизованной обработки запросов. Полезность централизации в системах обработки данных обычно обусловлена необходимостью управления разделением некоторого вычислительного ресурса (программ, данных, устройств или памяти) между многими пользователями. Примеры таких систем:

- администраторы устройств;
- файловые системы(локальные и распределенные);
- системы управления базами данных и т.п.

Внутренние механизмы операционных систем и прикладных систем, построенные по рассматриваемой схеме, как правило, ориентированы на обслуживание переменного во времени количества пользователей. Интерфейс связи удобно строить на основе механизма обмена сообщениями, рассмотренного ранее. При этом сервер обычно является задачей-получателем сообщений от задач пользователя, а обратная связь реализуется репликами ответов. Протокол передачи сообщений может быть любым. При работе в среде многозадачных ОС (QNX, UNIX, OS/2, Nowell Netware) средства его построения предоставляет система. Формально сервер как механизм синхронизации процессов играет роль монитора.

Разделяют два вида серверов:

- поддержка транзакций - обслуживание независимых запросов;
- поддержка зависимых запросов задачи.

Транзакция проще в программировании - каждый запрос рассматривается как новый, поэтому сервер не должен учитывать состояние задачи пользователя или следить за ее уничтожением. Однако пользователь обязан предоставлять в каждом запросе полное описание задачи.

Поддержка зависимых запросов задачи требует от нее начального запроса на открытие ресурса и конечного запроса на его закрытие. Открытие обычно использует символическое имя ресурса, которому сервером ставится в соответствие некоторый идентификатор ресурса для последующих обращений. Например, файловая система назначает при открытии имени файла указатель структуры описания файла с его номером в системной таблице (см. функции fopen, fileno, _open). Очевидно, чтосервер здесь должен быть информирован о фактах уничтожения задач, которые могут не закрывать ресурс.

**Пример построения системы "клиент-сервер" в QNX.**

Пусть в системе управления транспортными процессами возникает потребность определения множества справок по кратчайшим маршрутам между произвольными вершинами некоторой транспортной сети. Можно показать, что выдачу справок для реальных транспортных сетей эффектив- нее производить не

путем выборки из файла матрицы кратчайших маршрутов, а посредством оптимизации маршрута по запросу.

Задача поиска кратчайших маршрутов на графе транспортной сети в математическом отношении является хорошо изученной, а практический интерес в последнее время представляют лишь способы реализации вычислительных схем ее решения. Здесь мы будем использовать вариант эффективной реализации алгоритма Дийкстры. Таким образом, каждый запрос на оптимизацию маршрута интерпретируется в терминах получения характеристик дерева кратчайших маршрутов от некоторой исходной вершины до конкретно заданной вершины либо до всех остальных вершин графа сети. Взаимозависимостью запросов, ассоциированных при этом лишь с некоторой корневой вершиной, не будем пренебрегать, но для простоты рассуждений естественно возникающую идею диспетчеризации запросов оставим вне рассмотрения.

Реализация схемы решения любой задачи вынуждает выполнить предварительное планирование распределения памяти для представления переменных состояния (по крайней мере, с целью оценки реальной возможности решения задачи).

Область определения процесса оптимизации маршрутов на транспортной сети включает:

- множество данных представления графа транспортной сети;
- массивы результата оптимизации в виде описания дерева кратчайших маршрутов и расстояний от заданной корневой вершины;
- множество переменных описания инвариантных характеристик задачи оптимизации маршрутов на графе транспортной сети.

Рассмотрим последовательно состав перечисленных элементов с целью определения структуры глобальных переменных процедур оптимизации.

Графы реальных транспортных сетей обычно характеризуются незначительной степенью связности, поэтому их представление в памяти удобно выбирать в виде списковых структур данных (см. 4). Модель транспортной сети в этом случае включает следующие элементы данных:

N_dot - количество вершин графа транспортной сети;

N_lst - массив указателей списков смежности;

B_arc - массив элементов списков смежности (конечных вершин дуг графа);

W_arc - массив весов дуг.

Предполагается, что вершины графа транспортной сети пронумерованы числами 0,1,... N_dot-1.

Элементы массивов N_lst, B_arc, W_arc после нумерации вершин фиксируются следующим образом:

```
N_lst[0]=0;
```

N_lst[i+1]=N_lst[i]+card(i'), i=0,1,... N_dot-1;

B_arc[N_lst[i]+j]=i'(j),
W_arc[N_lst[i]+j]=w(i,i'(j)),
j=0,1,...card(i')-1, i=0,1,... N_dot-1.

Здесь x' - множество номеров вершин, для которых существуют направленные дуги из вершины графа x;

w(x,y) - вес дуги из вершины x в вершину y;

card(z) - размерность множества z.

Очевидно, что

card(N_lst)=N_dot+1,
card(B_arc)=card(W_arc)=N_lst[N_dot]

(элемент N_lst[N_dot] содержит количество дуг графа сети по определению).

Результат построения дерева кратчайших маршрутов размещается в следующих массивах:

D_dot - массив расстояний от корня дерева;

P_dot - массив вершин кратчайшего пути.

Размерность таких массивов

card(D_dot)=card(P_dot)=N_dot.

Состав инвариантных параметров задачи оптимизации маршрутов на графе транспортной сети, которые целесообразно определить один раз на этапе инициализации резидентной части программы, предопределен процедурой оптимизации маршрутов. Можно показать, что такими пара- метрами являются:

L_arc - максимальная длина дуги;

t_root - номер корневой вершины последнего полностью построенного дерева кратчайших маршрутов;

x_next и x_prec - массивы указателей следующих и предыдущих элементов списка очередей вершин;

t_size - размер горизонта планирования дерева;

t_base и t_stop - начало и конец индексов указателей очередей вершин.

Списки очередей вершин представимы в массивах следующей размер- ности [3]:

card(x_prec)=N_dot;
card(x_next)=N_dot+L_arc+1. (12.4)

Очевидно, что наиболее существенный вес здесь имеют области массивов.

Рассмотрим пример программной реализации в среде ОС QNX справочника кратчайших маршрутов задачей-администратором со следующей схемой функционирования:

- чтение параметров размерности и описания графа транспортной сети из файла сжатого описания сети;
- подключение идентификационного имени задачи-администратора;
- переход в состояние обработки запросов.

Предполагается, что файл сжатого описания сети подготовлен заранее с помощью загружаемой в пакетном режиме программы. Соответствие имени такого файла идентификационному имени задачи - администратора задается командной строкой запуска этой задачи:

```
interx netfile tskname
```

Здесь interx - имя файла - загрузочного модуля задачи - администратора (результата компиляции и компоновки исходного текста из файла interx.c);

netfile - имя файла сжатого описания транспортной сети;

tskname - идентификационное имя задачи - администратора. Задача - администратор обрабатывает следующие запросы:

- запрос операции расчета;
- запрос параметров сети;
- команда завершения работы.

Структуры сообщений потока информационного обмена и коды запросов определены в заголовочном файле interz.h.

Содержимое заголовочного файла interz.h: /* Декларация структур сообщений */

```
#ifndef INTERZ_H
#define INTERZ_H

typedef struct { /* Запрос на расчет маршрута */
 int code;       /* код сообщения */
 int sour;       /* номер исходной вершины */
 int dest;       /* номер конечной вершины */
} QUERY;

typedef struct { /* Запрос размерности описания графа сети */
 int code;       /* код сообщения */
 int dots;       /* количество вершин */
 int arcs;       /* количество дуг */
} CARDS;

typedef struct { /* Элемент результата расчета */
 int code;       /* код сообщения */
 int prec;       /* номер предшествующей вершины маршрута */
 long dist;      /* значение расстояния от корневой вершины */
} RESULT;
```

```
/* Коды сообщений потока межзадачного обмена */

enum {
 CALC=1, /* Запрос операции расчета */
 EXIT,   /* Команда завершения работы */
 SIZE    /* Запрос параметров сети */
};

#endif
```

Исходный текст задачи - администратора (размещается в файле interx.c):

```
#include <stdio.h>
#include <magic.h>
#include "interz.h"

int t_root;

/* Параметры описания графа сети */

int   N_dot;  /* количество вершин */
int   N_arc;  /* количество дуг */
int  *N_lst;  /* массив указателей списков смежности */
int  *B_arc;  /* массив конечных вершин дуг графа */
int  *W_arc;  /* массив весов дуг */

/* Параметры описания дерева кратчайших путей */

int *x_next;
int *x_prec;
int  t_size;
int  t_base;
int  t_stop;

/* Массивы результатов */

int  *P_dot;  /* массив вершин кратчайшего пути */
long *D_dot;  /* массив расстояний */

int mread(void *p, int s, int n, FILE *f) {
 return (fread(p,s,n,f)!=n);
}

void main(int na, char **la) {
 FILE *stream;
 QUERY buf;     /* Буфер приема запросов */
 RESULT out;    /* Буфер выдачи результатов */
```

```
 int L_arc;     /* Максимальная длина дуги */
 unsigned xid;  /* Идентификатор прикладной задачи */
 long length;

 if (na<3) {
  printf("\n\n %s","СПРАВОЧНИК КРАТЧАЙШИХ РАССТОЯНИЙ");
  printf("\n\nСинтаксис вызова:\n\n %s netfile tskname",la[0]);
  printf("\n   netname - имя файла сжатого описания сети");
  printf("\n   tskname - локальное имя задачи");
  exit(1);
 }
 if (!(stream=fopen(la[1],"r")))
  error("Ошибка открытия файла модели сети");

/* Чтение параметров размерности сети */

 if (
  mread(&N_dot,sizeof(N_dot),1,stream) ||
  mread(&N_arc,sizeof(N_arc),1,stream) ||
  mread(&L_arc,sizeof(L_arc),1,stream)
 ) error("Ошибка чтения размерности сети");

/* Захват памяти для рабочих массивов */

 t_base=N_dot+1, t_size=L_arc+1, t_stop=N_dot+t_size;
 length=(long)N_dot*(long)(sizeof(*D_dot));
 length+=(long)N_dot*(long)(sizeof(*P_dot)+sizeof(*x_prec));
 length+=(long)t_stop*(long)sizeof(*x_next);
 length+=(long)(N_dot+1)*(long)sizeof(*N_lst);
 length+=(long)N_arc*(long)(sizeof(*B_arc)+sizeof(*W_arc));
 x_next=malloc((unsigned)length);
 if (!x_next)
  error("Ошибка захвата памяти");

/* Инициализация указателей рабочих массивов */

 x_prec=x_next+t_stop;
 P_dot=x_prec+N_dot;
 N_lst=P_dot+N_dot;
 B_arc=N_lst+N_dot+1;
 W_arc=B_arc+N_arc;
 D_dot=(long *)(W_arc+N_arc);

/* Чтение массивов описания структуры сети */

 *N_lst=0, t_root=N_dot;
```

```
 if (
  mread(N_lst+1,sizeof(*N_lst),N_dot,stream) ||
  mread(B_arc,sizeof(*B_arc),N_arc,stream) ||
  mread(W_arc,sizeof(*W_arc),N_arc,stream)
 ) error("Ошибка чтения структуры сети");
 fclose(stream);

/* Переход в режим обслуживания запросов */

 if (name_attach(la[2],My_nid)) {
  while (N_dot)
   if ((xid=receive(0,&buf,sizeof(buf))) && (~xid))
    switch(buf.code) {
     case SIZE: /* Запрос параметров графа сети */
                buf.sour=N_dot;
                buf.dest=N_arc;
                reply(xid,&buf,sizeof(buf));
                break;

     case CALC: /* Заявка на оптимизацию маршрута */
                if (buf.sour!=t_root)
                 shrtpt(buf.sour,buf.dest);
                out.code=buf.code;
                out.dist=D_dot[buf.dest];
                out.prec=P_dot[buf.dest];
                reply(xid,&out,sizeof(out));
                break;

     case EXIT: /* Завершение работы */
                N_dot=0;
                break;
   }
 } else error("Ошибка подключения имени задачи");
}

/* Процедура построения кратчайших путей на графе */

void shrtpt(int i, int z) {
 int  j,k,l,m,n,w,t;
 long Q,R,S;
 static long infinity=0x7fffffffL;

/* Инициализация рабочих массивов */

 for (j=0; j<N_dot; D_dot[P_dot[j]=j]=infinity, j++);
```

```
 D_dot[x_next[t_root=N_dot]=i]=0, w=1;
 for (j=t_base; j<t_stop; x_next[j++]=N_dot);
 x_next[i]=x_prec[i]=N_dot;

/* Организация процесса ветвления */

 while (w>0) {
  for (m=N_dot; m<t_stop; m++)
   while((j=x_next[m])<N_dot) {
    if (j==z) return;
    k=x_next[m]=x_next[j], w--;
    if (k<N_dot) x_prec[k]=m;
    for (l=N_lst[j], Q=D_dot[j]; l<N_lst[j+1]; l++) {
     if ((S=D_dot[(n=B_arc[l])])>=Q)
      if ((R=Q+(t=W_arc[l]))<S) {
       if (S==infinity) w++;
       else {
        k=x_next[x_prec[n]]=x_next[n];
        if (k<N_dot) x_prec[k]=x_prec[n];
       }
       D_dot[n]=R, P_dot[n]=j, t+=m;
       if (t>=t_stop) t-=t_size;
       k=x_next[n]=x_next[t], x_prec[n]=t, x_next[t]=n;
       if (k<N_dot) x_prec[k]=n;
      }
    }
   }
 }
 t_root=i;
}
```

Прикладные задачи могут установить связь с задачей - администратором посредством проверки активности задачи - владельца идентификационного имени. После установления связи и получения параметров размерности описания графа транспортной сети можно порождать поток запросов на получение сведений о кратчайших маршрутах между любыми вершинами сети.

Исходный текст примера прикладной программы формирования запросов (размещается в файле intery.c):

```
/* Пример прикладной программы формирования запросов */

#include <stdio.h>
#include <magic.h>
#include "interz.h"

void main(na,la) int na; char **la; {
```

```
 unsigned xid;  /* Идентификатор задачи - получателя */
 QUERY buf;     /* Буфер приема запросов */
 RESULT out;    /* Буфер выдачи результатов */
 union buffer { /* Объединение буферов обмена */
  QUERY  buf;
  RESULT out;
 };

 if (na<2) {
  printf("\n\n          %s","ПРОВЕРКА СПРАВОЧНИКА КРАТЧАЙШИХ
РАССТОЯНИЙ");
  printf("\n\nСинтаксис вызова:\n\n %s tskname",la[0]);
  printf("\n   tskname - локальное имя задачи");
  exit(1);
 }
 if (xid=name_locate(la[1],0,sizeof(union buffer))) {

/* Запрос параметров сети */

  buf.code=SIZE;
  send(xid,&buf,&buf,sizeof(buf));
  printf("\n Размерность транспортной сети:");
  printf("\n  количество вершин %d",buf.sour);
  printf("\n  количество дуг %d",buf.dest);

/* Цикл рабочих запросов */

  buf.code=CALC;
  while(printf("\nОткуда, куда - ? "),
        scanf(" %d %d",&buf.sour,&buf.dest)) {
   send(xid,&buf,&out,sizeof(buf));
   if (out.code==buf.code)
    printf("\n Расстояние %ld через вершину %d",out.dist,out.prec);
  }

/* Завершение работы */

  buf.code=EXIT;
  send(xid,&buf,&out,sizeof(buf));
 } else error("\n СПРАВОЧНИК КРАТЧАЙШИХ РАССТОЯНИЙ ПАССИВЕН");
 printf("\n Конец работы !\n\n");
}
```

Пусть intery - имя файла загрузочного модуля прикладной задачи формирования запросов (результата компиляции и компоновки исходного текста из

файла intery.c). Взаимодействие с задачей - администратором interx реализуется лишь после указания при запуске программы intery в командной строке вида

```
intery tskname
```

идентификационного имени tskname, использованного при запуске задачи interx.