ИУ7-63Б, Авдейкина Валерия

Лаб. работа /proc (версия булочной с однопоточным сервером)

Время обслуживания клиента сервером в случае однопоточной версии программы измеряется на стороне клиента. Пример работы программы представлен на рисунке 1 (клиент 3 обслужился раньше, чем 1 и 2, так как они опоздали; аналогично с клиентом 7 – клиент 6 опоздал).

Рисунок 1: Пример работы программы

Листинг 1: Содержимое файла bakery.x

```
/*
* filename: bakery.x
* function: Define constants, non-standard data types and the calling
*                       process in remote calls
*/
struct BAKERY
{
        int num;
        int pid;
        int res;
};
program BAKERY_PROG
{
        version BAKERY_VER
        {
                struct BAKERY GETN(struct BAKERY) = 1;
                struct BAKERY WAIT(struct BAKERY) = 2;
                struct BAKERY PROC(struct BAKERY) = 3;
        } = 1; /* Version number = 1 */
} = 0x20000001; /* RPC program number */
```

Листинг 2: Исходный код сервера (файл bakery_server.c)

```c
/*
 * This is sample code generated by rpcgen.
 * These are only templates and you can use them
 * as a guideline for developing your own functions.
 */
#define _GNU_SOURCE
#define MAX_CLIENT 40
#include <stdio.h>
#include <pthread.h>
#include <stdlib.h>
#include <time.h>
#include <unistd.h>
#include <stdbool.h>
#include "bakery.h"
#define MY_TIMEOUT_SEC 1
struct thread_arg
{
        int pid;
        int num;
        int res;
};
bool choosing[MAX_CLIENT] = { 0 };
int number[MAX_CLIENT] = { 0 };
int curr_res = 'a';
int local_pid = 0;
int last_num = 0;
// получение номера
struct BAKERY *
getn_1_svc(struct BAKERY *argp, struct svc_req *rqstp)
{
    static struct BAKERY  result;

    int i = local_pid;
    local_pid++;
    choosing[i] = true;

    int max_n = 0;

    for (int j = 0; j < MAX_CLIENT; j++)
        if (number[j] > max_n)
            max_n = number[j];

    number[i] = max_n + 1;
    result.pid = i;
    result.num = number[i];
    choosing[i] = false;

    return &result;
}
```

```
50  // обслуживание (булочная)
51  struct BAKERY *
52  wait_1_svc(struct BAKERY *argp, struct svc_req *rqstp)
53  {
54      static struct BAKERY result;
55      int i = argp->pid;
56      result.pid = i;
57      result.num = argp->num;
58      time_t start, end;
59      for (int j = 0; j < MAX_CLIENT; j++)
60      {
61          while (choosing[j]);
62          /* Если клиент опоздал, то есть его номер меньше,
63             чем номер последнего обслуженного, выполняется
64             отказ в обслуживании */
65          if (last_num > number[i]) {
66              number[i] = 0;
67              result.res = '0';
68              return &result;
69          }
70          /* Если время ожидания превысило MY_TIMEOUT_SEC,
71             значит, клиенты с меньшим номером опоздали */
72          start = clock();
73          while ((number[j] > 0) && (number[j] < number[i] ||
74                                     (number[j] == number[i] && j < i)))
75          {
76              end = clock();
77              if ((end - start) / CLOCKS_PER_SEC > MY_TIMEOUT_SEC)
78                  break;
79          }
80      }
81      result.res = curr_res;
82      curr_res++;
83      last_num = number[i];
84      number[i] = 0;
85      /* Если нет клиентов, ожидающих обслуживания,
86         сбрасывается номер последнего обслуженного
87         клиента */
88      for (int j = 0; j < MAX_CLIENT; j++)
89          if (number[j] > 0)
90              return &result;
91      last_num = 0;
92      return &result;
93  }
94  // не используется в однопоточной версии
95  struct BAKERY *
96  proc_1_svc(struct BAKERY *argp, struct svc_req *rqstp)
97  {
98      return NULL;
99  }
```

Листинг 3: Исходный код клиента (файл bakery_client.c)

```c
/*
 * This is sample code generated by rpcgen.
 * These are only templates and you can use them
 * as a guideline for developing your own functions.
 */
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <time.h>
#include "bakery.h"

void
bakery_prog_1(char *host)
{
    CLIENT *clnt;
    struct BAKERY  *result_1;
    struct BAKERY  getn_1_arg;
    struct BAKERY  *result_2;
    struct BAKERY  proc_1_arg;
    struct BAKERY  *result_3;
    struct BAKERY  wait_1_arg;
    time_t time_numb, time_wait, time_serv;

    clnt = clnt_create (host, BAKERY_PROG, BAKERY_VER, "udp");
    if (clnt == NULL) {
        clnt_pcreateerror (host);
        exit (1);
    }

    srand(time(NULL));
    double sleep_time = (double)rand() / RAND_MAX * 1000000 * 1.5;
    usleep(sleep_time);

    time_numb = clock();
    result_1 = getn_1(&getn_1_arg, clnt);
    if (result_1 == (struct BAKERY *) NULL) {
        clnt_perror (clnt, "call failed");
    }
    time_numb = clock() - time_numb;
    printf("getn: client pid=%2d, num=%2d\n", result_1->pid, result_1→num);

    wait_1_arg.num = result_1->num;
    wait_1_arg.pid = result_1->pid;
    sleep(rand() % 7 + 5);
    time_wait = clock();
    result_2 = wait_1(&wait_1_arg, clnt);
    if (result_2->res == '0') {
        clnt_perror (clnt, "timeout!");
        clnt_destroy (clnt);
```

```
50          exit (1);
51      }
52      time_wait = clock() - time_wait;
53      time_serv = 0;
54
55      double t = (double) (time_numb + time_wait + time_serv) * 1000000 /
56  CLOCKS_PER_SEC;
57      printf("proc: client pid=%2d, res=%c (sleep=%.1fs)\ntotal_time=%.4fus\
58  n", result_2->pid, result_2->res, sleep_time / 1000000, t);
59      clnt_destroy (clnt);
60  }
61
62  int
63  main (int argc, char *argv[])
64  {
65      char *host;
66
67      if (argc < 2) {
68          printf ("usage: %s server_host\n", argv[0]);
69          exit (1);
70      }
71
72      host = argv[1];
73      bakery_prog_1 (host);
74      exit (0);
75  }
```

Листинг 4: bakery_clnt.c

```
/*
 * Please do not edit this file.
 * It was generated using rpcgen.
 */

#include <memory.h> /* for memset */
#include "bakery.h"

/* Default timeout can be changed using clnt_control() */
static struct timeval TIMEOUT = { 25, 0 };

struct BAKERY *
getn_1(struct BAKERY *argp, CLIENT *clnt)
{
        static struct BAKERY clnt_res;

        memset((char *)&clnt_res, 0, sizeof(clnt_res));
        if (clnt_call (clnt, GETN,
                (xdrproc_t) xdr_BAKERY, (caddr_t) argp,
                (xdrproc_t) xdr_BAKERY, (caddr_t) &clnt_res,
                TIMEOUT) != RPC_SUCCESS) {
                return (NULL);
```

```
        }
        return (&clnt_res);
}

struct BAKERY *
wait_1(struct BAKERY *argp, CLIENT *clnt)
{
        static struct BAKERY clnt_res;

        memset((char *)&clnt_res, 0, sizeof(clnt_res));
        if (clnt_call (clnt, WAIT,
                (xdrproc_t) xdr_BAKERY, (caddr_t) argp,
                (xdrproc_t) xdr_BAKERY, (caddr_t) &clnt_res,
                TIMEOUT) != RPC_SUCCESS) {
                return (NULL);
        }
        return (&clnt_res);
}

struct BAKERY *
proc_1(struct BAKERY *argp, CLIENT *clnt)
{
        static struct BAKERY clnt_res;

        memset((char *)&clnt_res, 0, sizeof(clnt_res));
        if (clnt_call (clnt, PROC,
                (xdrproc_t) xdr_BAKERY, (caddr_t) argp,
                (xdrproc_t) xdr_BAKERY, (caddr_t) &clnt_res,
                TIMEOUT) != RPC_SUCCESS) {
                return (NULL);
        }
        return (&clnt_res);
}
```

Листинг 5: bakery_svc.c

```
/*
 * Please do not edit this file.
 * It was generated using rpcgen.
 */

#include "bakery.h"
#include <stdio.h>
#include <stdlib.h>
#include <rpc/pmap_clnt.h>
#include <string.h>
#include <memory.h>
#include <sys/socket.h>
#include <netinet/in.h>

#ifndef SIG_PF
```

```
#define SIG_PF void(*)(int)
#endif

static void
bakery_prog_1(struct svc_req *rqstp, register SVCXPRT *transp)
{
	union {
		struct BAKERY getn_1_arg;
		struct BAKERY wait_1_arg;
		struct BAKERY proc_1_arg;
	} argument;
	char *result;
	xdrproc_t _xdr_argument, _xdr_result;
	char *(*local)(char *, struct svc_req *);

	switch (rqstp->rq_proc) {
	case NULLPROC:
		(void) svc_sendreply (transp, (xdrproc_t) xdr_void, (char *)NULL);
		return;

	case GETN:
		_xdr_argument = (xdrproc_t) xdr_BAKERY;
		_xdr_result = (xdrproc_t) xdr_BAKERY;
		local = (char *(*)(char *, struct svc_req *)) getn_1_svc;
		break;

	case WAIT:
		_xdr_argument = (xdrproc_t) xdr_BAKERY;
		_xdr_result = (xdrproc_t) xdr_BAKERY;
		local = (char *(*)(char *, struct svc_req *)) wait_1_svc;
		break;

	case PROC:
		_xdr_argument = (xdrproc_t) xdr_BAKERY;
		_xdr_result = (xdrproc_t) xdr_BAKERY;
		local = (char *(*)(char *, struct svc_req *)) proc_1_svc;
		break;

	default:
		svcerr_noproc (transp);
		return;
	}
	memset ((char *)&argument, 0, sizeof (argument));
	if (!svc_getargs (transp, (xdrproc_t) _xdr_argument, (caddr_t) &argument)) {
		svcerr_decode (transp);
		return;
	}
	result = (*local)((char *)&argument, rqstp);
	if (result != NULL && !svc_sendreply(transp, (xdrproc_t) _xdr_result, result)) {
		svcerr_systemerr (transp);
	}
	if (!svc_freeargs (transp, (xdrproc_t) _xdr_argument, (caddr_t) &argument)) {
```

```
                fprintf (stderr, "%s", "unable to free arguments");
                exit (1);
        }
        return;
}

int
main (int argc, char **argv)
{
        register SVCXPRT *transp;

        pmap_unset (BAKERY_PROG, BAKERY_VER);

        transp = svcudp_create(RPC_ANYSOCK);
        if (transp == NULL) {
                fprintf (stderr, "%s", "cannot create udp service.");
                exit(1);
        }
        if (!svc_register(transp, BAKERY_PROG, BAKERY_VER, bakery_prog_1,
IPPROTO_UDP)) {
                fprintf (stderr, "%s", "unable to register (BAKERY_PROG, BAKERY_VER,
udp).");
                exit(1);
        }

        transp = svctcp_create(RPC_ANYSOCK, 0, 0);
        if (transp == NULL) {
                fprintf (stderr, "%s", "cannot create tcp service.");
                exit(1);
        }
        if (!svc_register(transp, BAKERY_PROG, BAKERY_VER, bakery_prog_1,
IPPROTO_TCP)) {
                fprintf (stderr, "%s", "unable to register (BAKERY_PROG, BAKERY_VER,
tcp).");
                exit(1);
        }

        svc_run ();
        fprintf (stderr, "%s", "svc_run returned");
        exit (1);
        /* NOTREACHED */
}
```

Листинг 6: bakery_xdr

```
/*
 * Please do not edit this file.
 * It was generated using rpcgen.
 */

#include "bakery.h"
```

```
bool_t
xdr_BAKERY (XDR *xdrs, BAKERY *objp)
{
        register int32_t *buf;

         if (!xdr_int (xdrs, &objp->num))
                 return FALSE;
         if (!xdr_int (xdrs, &objp->pid))
                 return FALSE;
         if (!xdr_int (xdrs, &objp->res))
                 return FALSE;
        return TRUE;
}
```