```
/*
 * 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 <pthread.h>

#include "bakery.h"

#define N 50

int choosing[N] = { 0 };
int num[N] = { 0 };
int id = 0;
int data = 1;

struct arg_t
{
	int id;
	int num;
};

void get_num(struct arg_t *arg)
{
	sleep(rand() % 3);

	while (choosing[id]);

	choosing[id] = 1;
	int i = id;
	id++;
	arg->id = i;
	int max = 0;
	for (int j = 0; j < N; j++)
		if (num[j] > max)
			max = num[j];
	num[i] = max + 1;
	arg->num = num[i];
	choosing[i] = 0;
}

int lexical_less(int a1, int a2, int b1, int b2) {
	if (a1 < b1)
		return 1;
	if (a1 > b1)
		return 0;
	return a2 < b2;
}

int get_data(struct arg_t *arg)
{
	sleep(rand() % 3);

	//printf("--- %d\n", arg->id);

	int i = arg->id;
	//printf("1");
	for (int j = 0; j < N; j++)
	{
		while (choosing[j]) printf("2");
		while (num[j] != 0 && lexical_less(num[j], j, num[i], i)); //printf("3");
		printf("4");
	}
	int d = data;
	data++;
	num[i] = 0;
	//printf("OK");
	return d;
}

bool_t
bakery_proc_1_svc(struct BAKERY *argp, struct BAKERY *result, struct svc_req *rqstp)
{
	bool_t retval;

	struct arg_t arg;


	switch (argp->op)
	{
		case GET_NUM:
		{
			get_num(&arg);
			result->id = arg.id;
			result->num = arg.num;
			break;
		}
		case OPEN_CRIT_SECTION:
		{
			arg.id = argp->id;
			result->res = get_data(&arg);
			break;
		}
	}

	return retval;
}

int
bakery_prog_1_freeresult (SVCXPRT *transp, xdrproc_t xdr_result, caddr_t result)
{
	xdr_free (xdr_result, result);
	/*
	 * Insert additional freeing code here, if needed
	 */

	return 1;
}

?????????????????????????????????????????????????????

/*
 * 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 "bakery.h"


void
bakery_prog_1(char *host)
{
	CLIENT *clnt;
	enum clnt_stat retval_1;
	struct BAKERY result_1;
	struct BAKERY  bakery_proc_1_arg;

#ifndef	DEBUG
	clnt = clnt_create (host, BAKERY_PROG, BAKERY_VER, "udp");
	if (clnt == NULL) {
		clnt_pcreateerror (host);
		exit (1);
	}
#endif	/* DEBUG */

	bakery_proc_1_arg.op = GET_NUM;
	bakery_proc_1_arg.pid = getpid();
	//printf("my pid %d\n", getpid());

	retval_1 = bakery_proc_1(&bakery_proc_1_arg, &result_1, clnt);
	if (retval_1 != RPC_SUCCESS) {
		clnt_perror (clnt, "call failed");
	}

	printf("received number from server: %d\n", result_1.num);

	sleep(rand() % 3);

	bakery_proc_1_arg.op = OPEN_CRIT_SECTION;
	bakery_proc_1_arg.id = result_1.id;
	bakery_proc_1_arg.pid = getpid();
	retval_1 = bakery_proc_1(&bakery_proc_1_arg, &result_1, clnt);
	if (retval_1 != RPC_SUCCESS) {
		clnt_perror (clnt, "call failed");
	}

	printf("received value from server: %d\n", result_1.res);
#ifndef	DEBUG
	clnt_destroy (clnt);
#endif	 /* DEBUG */
}


int
main (int argc, char *argv[])
{
	char *host;

	if (argc < 2) {
		printf ("usage: %s server_host\n", argv[0]);
		exit (1);
	}
	host = argv[1];
	bakery_prog_1 (host);
exit (0);
}
```