```
pthread_t p_thread;
pthread_attr_t attr;
struct thr_data{
    struct svc_req *rqstp;
    SVCXPRT *transp;
};
void* serv_request(void *data){
    struct thr_data *ptr_data;
    union {
        int service_1_arg;
    } argument;
    union {
        char service_1_res;
    } result;
    bool_t retval;
    xdrproc_t _xdr_argument, _xdr_result;
    bool_t (*local)(char *, void *, struct svc_req *);
    ptr_data = (struct thr_data  *)data;
    struct svc_req *rqstp = ptr_data->rqstp;
    register SVCXPRT *transp = ptr_data->transp;
    switch (rqstp->rq_proc) {
    case NULLPROC:
        (void) svc_sendreply (transp, (xdrproc_t) xdr_void, (char *)NULL);
        pthread_exit(0);
    case SERVICE:
        _xdr_argument = (xdrproc_t) xdr_int;
        _xdr_result = (xdrproc_t) xdr_char;
        local = (bool_t (*) (char *, void *,  struct svc_req *))service_1_svc;
        break;
    default:
        pthread_exit(0);
    }
    memset ((char *)&argument, 0, sizeof (argument));
    if (!svc_getargs (transp, (xdrproc_t) _xdr_argument, (caddr_t) &argument))
{
        svcerr_decode (transp);
        pthread_exit(0);
    }
    retval = (bool_t) (*local)((char *)&argument, (void *)&result, rqstp);
    if (retval > 0 && !svc_sendreply(transp, (xdrproc_t) _xdr_result, (char
*)&result)) {
        svcerr_systemerr (transp);
    }
}
static void pc_prog_1(struct svc_req *rqstp, register SVCXPRT *transp){
    struct thr_data *data_ptr = (struct thr_data*) malloc(sizeof(struct
thr_data));
    data_ptr->rqstp = rqstp;
    data_ptr->transp = transp;
    pthread_attr_setdetachstate(&attr,PTHREAD_CREATE_DETACHED);
    if(pthread_create(&p_thread, &attr, serv_request, (void *)data_ptr) != 0) {
        perror("pthread_create");
        exit(1);
   }
```