25 Mars 1992

Examen DESS

de Génie Logiciel

UNIX-C

durée: 2 heures
tout document autorisé

Notation:	Exercices :	1.	a.	1 point
			b.	1/2 point
			c.	1 point
		2.	a.	1/2 point
			b.	1/2 point
			c.	1/2 point
		3.	a.	1 point
			b.i	1/2 point
			b.ii	1/2 point
	Problème :	1.	a.	1/2 point
			b.	3 points
		2.	a.	1 points
			b.	3 points
			c.	2.5 points
			d.	4 points

Exercices

1. X Window
   (a) Donnez le schèma (algorithme) classique d'un client X Window.
   (b) On vous demande de porter X Window sur une machine donnée. Quel est théoriquement le travail à faire ?
   (c) On imagine une application (un client X) gèrant la souris. Expliquez le cheminement des informations relatives à la souris entre les différentes entités (physiques et logicielles) concernées. Un schèma vaut parfois mieux qu'un long discours ...

2. Communication inter-processus
   (a) Peut on faire communiquer par pipe (au sens BSD, c'est à dire celui vu en cours) des processus s'exécutant sur des machines différentes. Pourquoi ?
   (b) Quelle sont les solutions possibles si les machines partagent un système de fichiers ?
   (c) Même question que (b) si elles ne partagent pas de système de fichiers

3. Divers
   (a) Ecrire un script permettant de compresser tous les .o de l'arborescence dont la racine est le répertoire d'accueil d'un utilisateur donné.
   (b) On considère la fonction GnuLisp suivante:

```
(defun merge (f1 f2)
  (progn
    (insert-file f1)
    (insert-file f2)
  )
)
```

Questions

i. La rendre interactive
ii. Qu'est ce que cela apporte du point de vue de l'utilisateur ?

Problème

1. RPC
   (a) Que signifie RPC ?
   (b) Quelle est la différence entre RPC et dialogue via des sockets entre une application et un démon ? Faire un schèma comparatif.
2. PORTMAP
   (a) Quel est le rôle de portmap ?
   (b) Donner le schèma des échanges d'information lors du RPC mount; on s'intéresse en particulier aux processus et fichiers utilisés sur la machine locale et sur la machine distante.
   (c) Comment trouve-t-on le numéro correspondant à un RPC, connaissant son nom ? Ecrire un programme prenant en paramètre un mon de RPC et qui retourne le numéro correspondant.
   (d) Ecrire l'équivalent de la commande `rpcinfo -p` $<$ `hostname` $>$ sachant que l'on vous donne la fonction de l'annexe VIII et le fichier .h de l'annexe IX.

Annexe I: man 3 RPC

```
NAME
     rpc - library routines for remote procedure calls

SYNOPSIS AND DESCRIPTION
     RPC routines allow C programs to  make  procedure  calls  on
     other  machines across the network.  First, the client calls
     a procedure to send a request to the server.   Upon  receipt
     of  the request, the server calls a dispatch routine to per-
     form the requested service, and then  sends  back  a  reply.
     Finally, the procedure call returns to the client.

     All RPC  routines  require  the  header  <rpc/rpc.h>  to  be
     included.

     The RPC routines have been grouped by usage on the following
     man pages.

     portmap(3N)        Library routines for the RPC  bind  ser-
                        vice,  portmap(8C).   The routines docu-
                        mented on this page include:
                             pmap_getmaps()
                             pmap_getport()
                             pmap_rmtcall()
                             pmap_set()
                             pmap_unset()
                             xdr_pmap()
                             xdr_pmaplist()

     rpc_clnt_auth(3N)  Library routines for client side  remote
                        procedure call authentication.  The rou-
                        tines documented on this page include:
                             auth_destroy()
                             authnone_create()
                             authunix_create()
                             authunix_create_default()

     rpc_clnt_calls(3N) Library routines for client side  calls.
                        The  routines  documented  on  this page
                        include:
                             callrpc()
                             clnt_broadcast()
                             clnt_call()
                             clnt_freeres()
                             clnt_geterr()
                             clnt_perrno()
                             clnt_perror()
                             clnt_sperrno()
                             clnt_sperror()

     rpc_clnt_create(3N) Library routines for  dealing  with  the
                        creation and manipulation of CLIENT han-
                        dles.  The routines documented  on  this

                        page include:
```

```
                     clnt_control()
                     clnt_create()
                     clnt_create_vers()
                     clnt_destroy()
                     clnt_pcreateerror()
                     clntraw_create()
                     clnt_spcreateerror()
                     clnttcp_create()
                     clntudp_bufcreate()
                     clntudp_create()
                     rpc_createrr()

rpc_svc_calls(3N)  Library routines for registerring
                   servers.  The routines documented on
                   this page include:
                     registerrpc()
                     svc_register()
                     svc_unregister()
                     xprt_register()
                     xprt_unregister()

rpc_svc_create(3N) Library routines for dealing with the
                   creation of server side handles. The
                   routines documented on this page
                   include:
                     svc_destroy()
                     svcfd_create()
                     svcraw_create()
                     svctcp_create()
                     svcudp_bufcreate()

rpc_svc_err(3N)    Library routines for server side remote
                   procedure call errors.  The routines
                   documented on this page include:
                     svcerr_auth()
                     svcerr_decode()
                     svcerr_noproc()
                     svcerr_noprog()
                     svcerr_progvers()
                     svcerr_systemerr()
                     svcerr_weakauth()

rpc_svc_reg(3N)    Library routines for RPC servers.  The
                   routines documented on this page
                   include:
                     svc_fds()
                     svc_fdset()
                     svc_freeargs()
                     svc_getargs()
                     svc_getcaller()
                     svc_getreq()

                     svc_getreqset()
                     svc_run()
                     svc_sendreply()

rpc_xdr(3N)        XDR library routines for remote pro-
```

cedure calls. The routines documented on this page include:

- xdr_accepted_reply()
- xdr_authunix_parms()
- xdr_callhdr()
- xdr_callmsg()
- xdr_opaque_auth()
- xdr_rejected_reply()
- xdr_replymsg()

secure_rpc(3N) Library routines for secure remote procedure calls. The routines documented on this page include:

- authdes_create()
- authdes_getucred()
- get_mayaddress()
- getnetname()
- host2netname()
- key_decryptsession()
- key_encryptsession()
- key_gendes()
- key_setsecret()
- netname2host()
- netname2user()
- user2netname()

SEE ALSO

portmap(3N), rpc_clnt_auth(3N), rpc_clnt_calls(3N), rpc_clnt_create(3N), rpc_svc_calls(3N), rpc_svc_create(3N), rpc_svc_err(3N), rpc_svc_reg(3N), rpc_xdr(3N), secure_rpc(3N), xdr(3N), publickey(5), portmap(8C), keyserv(8C)

Network Programming

Annexe II: man 8 portmap

```
NAME
     portmap - TCP/IP port to RPC program number mapper

SYNOPSIS
     /usr/etc/portmap

DESCRIPTION
     portmap is a  server  that  converts  TCP/IP  protocol  port
     numbers  into  RPC  program  numbers.  It must be running in
     order to make RPC calls.

     When an RPC server is started, it  will  tell  portmap  what
     port number it is listening to, and what RPC program numbers
     it is prepared to serve.  When a client wishes  to  make  an
     RPC  call  to  a given program number, it will first contact
     portmap on the server machine to determine the  port  number
     where RPC packets should be sent.

     Normally, standard RPC servers are started by inetd(8C),  so
     portmap must be started before inetd is invoked.

SEE ALSO
     inetd.conf(5), inetd(8C), rpcinfo(8C)

BUGS
     If portmap crashes, all servers must be restarted.
```

Annexe III: man 5 rpc

```
NAME
     rpc - rpc program number data base

SYNOPSIS
     /etc/rpc

DESCRIPTION
     The rpc file contains user readable names that can  be  used
     in  place of rpc program numbers.  Each line has the follow-
     ing information:

          rpc-program-server  rpc-program-number  aliases

     Items are separated by any number of blanks and/or tab char-
     acters.  A ``#'' indicates the beginning of a comment; char-
     acters up to the end of the line are not interpreted by rou-
     tines which search the file.

EXAMPLE
     Here is an example of the /etc/rpc file from the SunOS  Sys-
     tem.
           #
           #            rpc   1.10   87/04/10
           #
           portmapper      100000   portmap sunrpc
           rstatd          100001   rstat rup perfmeter
           rusersd         100002   rusers
           nfs             100003   nfsprog
           ypserv          100004   ypprog
           mountd          100005   mount showmount
           ypbind          100007
           walld           100008   rwall shutdown
           yppasswdd       100009   yppasswd
           etherstatd      100010   etherstat
           rquotad         100011   rquotaprog quota rquota
           sprayd          100012   spray
           3270_mapper     100013
           rje_mapper      100014
           selection_svc   100015   selnsvc
           database_svc    100016
           rexd            100017   rex
           alis            100018
           sched           100019
           llockmgr        100020
           nlockmgr        100021
           x25.inr         100022
           statmon         100023
           status          100024
           bootparam       100026
           ypupdated       100028   ypupdate
           keyserv         100029   keyserver

FILES
     /etc/rpc
```

SEE ALSO

getrpcent(3N)

Annexe IV: man getrpcent

```
NAME
     getrpcent, getrpcbyname, getrpcbynumber,  endrpcent,  setrp-
     cent - get RPC entry

SYNOPSIS
     #include <netdb.h>

     struct rpcent *getrpcent()

     struct rpcent *getrpcbyname(name)
     char *name;

     struct rpcent *getrpcbynumber(number)
     int number;

     setrpcent (stayopen)
     int stayopen

     endrpcent ()

DESCRIPTION
     getrpcent, getrpcbyname, and getrpcbynumber() each return  a
     pointer to an object with the following structure containing
     the broken-out fields of a line in the  rpc  program  number
     data base, /etc/rpc.

          struct    rpcent {
               char *r_name;  /* name of server for this rpc program */
               char **r_aliases;   /* alias list */
               long r_number; /* rpc program number */
          };

     The members of this structure are:
          r_name              The name of the server for this rpc
                              program.
          r_aliases           A zero terminated list of alternate
                              names for the rpc program.
          r_number            The rpc  program  number  for  this
                              service.

     getrpcent() reads the next line of  the  file,  opening  the
     file if necessary.

     setrpcent() opens and rewinds the  file.   If  the  stayopen
     flag is non-zero, the net data base will not be closed after
     each call to getrpcent()  (either  directly,  or  indirectly
     through one of the other "getrpc" calls).

     endrpcent closes the file.

     getrpcbyname() and getrpcbynumber() sequentially search from
     the  beginning of the file until a matching rpc program name

     or program number is found, or until end-of-file is  encoun-
     tered.
```

FILES

/etc/rpc

SEE ALSO

rpc(5), rpcinfo(8C), ypserv(8)

DIAGNOSTICS

A NULL pointer is returned on EOF or error.

BUGS

All information is contained in a static area so it must be copied if it is to be saved.

Annexe V: man 8 rpcinfo

```
NAME
     rpcinfo - report RPC information

SYNOPSIS
     rpcinfo -p [ host ]

     rpcinfo [ -n portnum ] -u host program [ version ]

     rpcinfo [ -n portnum ] -t host program [ version ]

     rpcinfo -b program version

     rpcinfo -d program version

AVAILABILITY
     This program  is  available  with  the  Networking  software
     installation  option.   Refer  to  Installing  SunOS 4.1 for
     information on how to install optional software.

DESCRIPTION
     rpcinfo makes an RPC call to an RPC server and reports  what
     it finds.

OPTIONS
     -p   Probe the portmapper on host, and print a list  of  all
          registered  RPC programs.  If host is not specified, it
          defaults to the value returned by hostname(1).

     -u   Make an RPC call to  procedure  0  of  program  on  the
          specified host using UDP, and report whether a response
          was received.

     -t   Make an RPC call to  procedure  0  of  program  on  the
          specified host using TCP, and report whether a response
          was received.

     -n   Use portnum as the  port  number  for  the  -t  and  -u
          options  instead  of the port number given by the port-
          mapper.

     -b   Make an RPC broadcast to procedure 0 of  the  specified
          program and version using UDP and report all hosts that
          respond.

     -d   Delete registration for the RPC service of  the  speci-
          fied program and version.  This option can be exercised
          only by the super-user.

     The program argument can be either a name or a number.

     If a version is specified, rpcinfo  attempts  to  call  that
     version   of  the  specified  program.   Otherwise,  rpcinfo

     attempts to find all the registered version numbers for  the
```

specified program by calling version 0 (which is presumed not to exist; if it does exist, rpcinfo attempts to obtain this information by calling an extremely high version number instead) and attempts to call each registered version. Note: the version number is required for -b and -d options.

EXAMPLES

To show all of the RPC services registered on the local machine use:

```
example% rpcinfo -p
```

To show all of the RPC services registered on the machine named klaxon use:

```
example% rpcinfo -p klaxon
```

To show all machines on the local net that are running the Network Information Service (NIS) use:

```
example% rpcinfo -b ypserv 'version' | uniq
```

where 'version' is the current NIS version obtained from the results of the -p switch above.

To delete the registration for version 1 of the walld service use:

```
example% rpcinfo -d walld 1
```

SEE ALSO

rpc(5), portmap(8C)

RPC Programming Guide in Network Programming

BUGS

In releases prior to the SunOS 3.0 release, the Network File System (NFS) did not register itself with the portmapper; rpcinfo cannot be used to make RPC calls to the NFS server on hosts running such releases.

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

Annexe VI: utilisation de rpcinfo

```
/usr/etc/rpcinfo -p horus
   program vers proto   port
    100000    2   tcp    111  portmapper
    100000    2   udp    111  portmapper
    100007    2   tcp   1024  ypbind
    100007    2   udp   1027  ypbind
    100007    1   tcp   1024  ypbind
    100007    1   udp   1027  ypbind
    100029    1   udp    654  keyserv
    100005    1   udp    693  mountd
    100005    2   udp    693  mountd
    100005    1   tcp    696  mountd
    100005    2   tcp    696  mountd
    100003    2   udp   2049  nfs
    100024    1   udp    706  status
    100024    1   tcp    708  status
    100021    1   tcp    709  nlockmgr
    100021    1   udp   1034  nlockmgr
    100021    3   tcp    713  nlockmgr
    100021    3   udp   1035  nlockmgr
    100020    1   udp   1036  llockmgr
    100020    1   tcp    718  llockmgr
    100021    2   tcp    721  nlockmgr
    100021    2   udp   1037  nlockmgr
    100017    1   tcp   1026  rexd
    100011    1   udp   1038  rquotad
    100002    1   udp   1040  rusersd
    100002    2   udp   1040  rusersd
    100012    1   udp   1041  sprayd
    100008    1   udp   1042  walld
    300019    1   udp   1022  amd
```

Annexe VII: man 3 portmap

NAME

pmap_getmaps, pmap_getport, pmap_rmtcall, pmap_set, pmap_unset, xdr_pamp, xdr_pmaplist - library routines for RPC bind service

DESCRIPTION

These routines allow client C programs to make procedure calls to the RPC binder service. portmap(1) maintains a list of mappings between programs and their universal addresses.

Routines

```
#include <rpc/rpc.h>

struct pmaplist * pmap_getmaps(addr)
struct sockaddr_in *addr;
```

Return a list of the current RPC program-to-address mappings on the host located at IP address *addr. This routine returns NULL if the remote portmap service could not be contacted. The command 'rpcinfo -p' uses this routine (see rpcinfo(8C)).

```
u_short pmap_getport(addr, prognum, versnum, protocol)
struct sockaddr_in *addr;
u_long prognum, versnum, protocol;
```

Return the port number on which waits a service that supports program number prognum, version versnum, and speaks the transport protocol protocol. The address is returned in addr, which should be preallocated. The value of protocol can be either IPPROTO_UDP or IPPROTO_TCP. A return value of zero means that the mapping does not exist or that the RPC system failed to contact the remote portmap service. In the latter case, the global variable rpc_createeer (see rpc_clnt_create(3N)) contains the RPC status. If the requested version number is not registered, but at least a version number is registered for the given program number, the call returns a port number. Note: pmap_getport() returns the port number in host byte order. Some other network routines may require the port number in network byte order. For example, if the port number is used as part of the sockaddr_in structure, then it should be converted to network byte order using htons(3N).

```
enum clnt_stat pmap_rmtcall(addr, prognum, versnum, procnum, inproc, in, outproc, out, timeout, po
struct sockaddr_in *addr;
u_long prognum, versnum, procnum;
char *in, *out;
xdrproc_t inproc, outproc;
```

```
struct timeval timeout;
u_long *portp;
```

Request that the portmap on the host at IP address *addr make an RPC on the behalf of the caller to a procedure on that host. *portp is modified to the program's port number if the procedure succeeds. The definitions of other parameters are discussed in callrpc() and clnt_call() (see rpc_clnt_calls(3N)).

Warning: If the requested remote procedure is not registered with the remote portmap then no error response is returned and the call times out. Also, no authentication is done.

```
bool_t pmap_set(prognum, versnum, protocol, port)
u_long prognum, versnum;
int protocol;
u_short port;
```

Registers a mapping between the triple [prognum,versnum,protocol] and port on the local machine's portmap service. The value of protocol can be either IPPROTO_UDP or IPPROTO_TCP. This routine returns TRUE if it succeeds, FALSE otherwise. It is called by servers to register themselves with the local portmap. Automatically done by svc_register().

```
bool_t pmap_unset(prognum, versnum)
u_long prognum, versnum;
```

Deregisters all mappings between the triple [prognum,versnum,*] and ports on the local machine's portmap service. It is called by servers to deregister themselves with the local portmap. This routine returns TRUE if it succeeds, FALSE otherwise.

```
bool_t xdr_pmap(xdrs, regp)
XDR *xdrs;
struct pmap *regp;
```

Used for creating parameters to various portmap procedures, externally. This routine is useful for users who wish to generate these parameters without using the pmap interface. This routine returns TRUE if it succeeds, FALSE otherwise.

```
bool_t xdr_pmaplist(xdrs, rp)
XDR *xdrs;
struct pmaplist **rp;
```

Used for creating a list of port mappings, externally.

This routine is useful for users who wish to generate these parameters without using the pmap interface. This routine returns TRUE if it succeeds, FALSE otherwise.

SEE ALSO

rpc(3N), portmap(8C), rpcinfo(8C)

Annexe VIII: fonction get_ip_addr

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netdb.h>
#include <netinet/in.h>
#include <arpa/inet.h>


 get_ip_addr(machine,ip_addr)
/****************************/

 char *machine;
 struct in_addr *ip_addr;
{

 struct hostent *hp;

 if ( !(hp = gethostbyname(machine)) )
   {
    printf("Unknown host %s\n",machine);
    exit(-1);
   }
 bcopy(hp->h_addr,ip_addr,hp->h_length);
}
```

Annexe IX: fichier pmap_proto.h

```
/* @(#)pmap_prot.h 1.16 89/07/26 SMI */

/*
 * pmap_prot.h
 * Protocol for the local binder service, or pmap.
 *
 * Copyright (C) 1984, Sun Microsystems, Inc.
 *
 * The following procedures are supported by the protocol:
 *
 * PMAPPROC_NULL() returns ()
 *  takes nothing, returns nothing
 *
 * PMAPPROC_SET(struct pmap) returns (bool_t)
 *  TRUE is success, FALSE is failure.  Registers the tuple
 * [prog, vers, prot, port].
 *
 * PMAPPROC_UNSET(struct pmap) returns (bool_t)
 * TRUE is success, FALSE is failure.  Un-registers pair
 * [prog, vers].  prot and port are ignored.
 *
 * PMAPPROC_GETPORT(struct pmap) returns (long unsigned).
 * 0 is failure.  Otherwise returns the port number where the pair
 * [prog, vers] is registered.  It may lie!
 *
 * PMAPPROC_DUMP() RETURNS (struct pmaplist *)
 *
 * PMAPPROC_CALLIT(unsigned, unsigned, unsigned, string<>)
 *  RETURNS (port, string<>);
 * usage: encapsulatedresults =
 * PMAPPROC_CALLIT(prog, vers, proc, encapsulatedargs);
 *  Calls the procedure on the local machine.  If it is not registered,
 * this procedure is quite; ie it does not return error information!!!
 * This procedure only is supported on rpc/udp and calls via
 * rpc/udp.  This routine only passes null authentication parameters.
 * This file has no interface to xdr routines for PMAPPROC_CALLIT.
 *
 * The service supports remote procedure calls on udp/ip or tcp/ip socket 111.
 */

#ifndef _rpc_pmap_prot_h
#define _rpc_pmap_prot_h

#define PMAPPORT ((u_short)111)
#define PMAPPROG ((u_long)100000)
#define PMAPVERS ((u_long)2)
#define PMAPVERS_PROTO ((u_long)2)
#define PMAPVERS_ORIG ((u_long)1)
#define PMAPPROC_NULL ((u_long)0)
#define PMAPPROC_SET ((u_long)1)
#define PMAPPROC_UNSET ((u_long)2)
#define PMAPPROC_GETPORT ((u_long)3)
#define PMAPPROC_DUMP ((u_long)4)
#define PMAPPROC_CALLIT ((u_long)5)
```

```
struct pmap {
long unsigned pm_prog;
long unsigned pm_vers;
long unsigned pm_prot;
long unsigned pm_port;
};

extern bool_t xdr_pmap();

struct pmaplist {
struct pmap pml_map;
struct pmaplist *pml_next;
};

#ifndef KERNEL
extern bool_t xdr_pmaplist();
#endif /*!KERNEL*/

#endif /*!_rpc_pmap_prot_h*/
```