Mise en œuvre du protocole ICMP - raw sockets

type (8 ou 0)	code (0)	checksum
identificateur (pid sous UNIX)		n° de séquence
données optionnelles		

- **type : ICMP_ECHO (8) ou ICMP_ECHOREPLY (0)**
- **checksum : header ICMP + données (algorithme checksum IP)**
- **identificateur : ICMP étant au niveau 3 il n'existe pas de n° de port -> utilisation du pid du processus pour identifier à qui le paquet est destiné.**
- **n° de séquence : permet de distinguer les paquets**
- **données optionnelles : données pour le test (par exemple heure d'envoi)**

Fichiers header :

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

Structure ICMP :

```
struct icmp
  {
  u_char   icmp_type;
  u_char   icmp_code;
  u_short  icmp_cksum;
  u_short  icmp_id;
  u_short  icmp_seq;
  char     icmp_data[1];
  };
```

Réservation d'une zone mémoire pour les données ICMP :

```
#define MAX_PACKET 100

struct  icmp * icmp_buf;
char    pkt_buf[MAX_PACKET];

icmp_buf = (struct icmp *) pkt_buf;
```

Création d'un socket pour ICMP :

```
int  icmp_sock;

icmp_sock=socket(AF_INET, SOCK_RAW, IPPROTO_ICMP)
```

Initialisation de la structure contenant l'adresse :

```
struct sockaddr_in icmp_sock_info;
struct in_addr addr;
int lsock = sizeof(icmp_sock_info);

icmp_sock_info.sin_family= AF_INET;
inet_aton("127.0.0.1", &addr);
icmp_sock_info.sin_addr.s_addr=addr.s_addr;
```

Construction de la trame ICMP

```
#define IP_SIZE 20
#define ICMP_SIZE 16

int nseq = 0;

icmp_buf=(struct icmp *) pkt_buf;
icmp_buf->icmp_type=ICMP_ECHO;
icmp_buf->icmp_code=0;
icmp_buf->icmp_id=getpid();
icmp_buf->icmp_seq=nseq++;
gettimeofday((struct timeval *) icmp_buf->icmp_data, NULL);
icmp_buf->icmp_cksum=0;
icmp_buf->icmp_cksum=cksum((unsigned short *)  icmp_buf, ICMP_SIZE);
```

Les données optionnelles sont l'heure de l'envoi du paquet (obtenue par la fonction gettimeofday( )) - 8 octets

include <sys/time.h>

int gettimeofday(struct timeval *tp, NULL);

struct timeval
{
time_t   tv_sec seconds
suseconds_t tv_usec microseconds
}

chksum est une fonction qui calcule le checksum

Envoi de la trame ICMP

```
sendto(icmp_sock,icmp_buf,ICMP_SIZE,0,&icmp_sock_info, lsock);
```

Réception d'une trame ICMP

```
recvfrom(icmp_sock, pkt_buf, IP_SIZE + ICMP_SIZE,0, NULL, NULL);
```

La réponse contient également l'entête IP.

Pour récupérer la partie ICMP du paquet :

memcpy(icmp_buf, &pkt_buf[IP_SIZE], ICMP_SIZE);

Mini - ping.

```
/* file miniping.c

   mise en oeuvre du protocole ICMP pour tester si une adresse
   IP répond - mini programme ping - l'adresse IP est passée sur
   la ligne de commande

   Ph. CAMUS
   27/9/2000
*/

#include <stdio.h>
#include <unistd.h>
#include <sys/time.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netinet/ip_icmp.h>
#include <arpa/inet.h>

unsigned short cksum(unsigned short * addr, int len);
void tv_sub(struct timeval *out, struct timeval *in);

#define IP_SIZE 20
#define ICMP_SIZE 16 /* entête 8 octets, données 8 octets */
#define MAX_PACKET 100

int main(int argc, char * argv[])
{
int  icmp_sock;
struct sockaddr_in icmp_sock_info;
struct in_addr addr;
int lsock = sizeof(icmp_sock_info);
struct icmp * icmp_buf;
char pkt_buf[MAX_PACKET];
struct timeval tvrec;
double dif;

/* création d'un socket raw utilisant ICMP */
if ((icmp_sock=socket(AF_INET, SOCK_RAW, IPPROTO_ICMP))==-1)
 {
  perror ("Erreur de création du socket");
  exit (1);
 }

/* initialisation de la structure "adresse" du socket */
/* elle contient le type de socket, l'adresse IP mais pas de numéro de port car on est au
niveau 3*/
icmp_sock_info.sin_family= AF_INET;
inet_aton(argv[1], &addr);
icmp_sock_info.sin_addr.s_addr=addr.s_addr;
```

```
/* construction de la trame ICMP */
icmp_buf=(struct icmp *) pkt_buf;
icmp_buf->icmp_type=ICMP_ECHO;
icmp_buf->icmp_code=0;
icmp_buf->icmp_id=getpid();
icmp_buf->icmp_seq=0;
gettimeofday((struct timeval *) icmp_buf->icmp_data, NULL);
icmp_buf->icmp_cksum=0;
icmp_buf->icmp_cksum=cksum((unsigned short *)  icmp_buf, ICMP_SIZE);

printf("\nTrame ICMP envoyée ");
printf("\nType : %02X ", icmp_buf->icmp_type);
printf("ID : %04X ", icmp_buf->icmp_id);
printf("Cksum : %04X \n", icmp_buf->icmp_cksum);


/* envoie la trame  ICMP */
sendto(icmp_sock,icmp_buf,ICMP_SIZE,0,&icmp_sock_info, lsock);


/* réception de la réponse ICMP avec l'entête IP*/
do
  {
  recvfrom(icmp_sock, pkt_buf, IP_SIZE + ICMP_SIZE,0, NULL, NULL);
  memcpy(icmp_buf, &pkt_buf[IP_SIZE], ICMP_SIZE);
  }
while(icmp_buf->icmp_type != ICMP_ECHOREPLY);

gettimeofday((struct timeval *) &tvrec, NULL);
tv_sub(&tvrec, (struct timeval *) icmp_buf->icmp_data);
dif=tvrec.tv_sec + ((double)tvrec.tv_usec / 1000000L);

printf("\nTrame ICMP reçue après %.3f sec", dif);
printf("\nType : %02X ", icmp_buf->icmp_type);
printf("ID : %04X ", icmp_buf->icmp_id);
printf("Cksum : %04X \n", icmp_buf->icmp_cksum);

exit (0);
}
```

```
/* calcul de la somme de contrôle */
unsigned short cksum(unsigned short * addr, int len)

{
int nleft = len;
int sum = 0;
unsigned short *w = addr;
unsigned short answer = 0;

while (nleft >1)
  {
  sum = sum + *w++;
  nleft = nleft - 2;
  }

if (nleft == 1)
  {
  *(unsigned char *) (&answer) = * (unsigned char *) w;
  sum = sum + answer;
  }

sum = (sum >> 16) + (sum & 0xffff);
sum = sum + (sum >> 16);
answer = ~sum;

return answer;
}



/* calcul de la différence de deux valeurs timeval */
void tv_sub(struct timeval *out, struct timeval *in)
{
out->tv_usec=out->tv_usec - in->tv_usec;
if (out->tv_usec < 0)
  {
  out->tv_sec--;
  out->tv_usec = out->tv_usec + 1000000;
  }
out->tv_sec=out->tv_sec - in->tv_sec;
}
```

Résultat (sur 127.0.0.1) :

```
17:33:30.645412 localhost > localhost: icmp: echo request
                  4500 0024 0022 0000 4001 7cb5 7f00 0001
                  7f00 0001 0800 d9db d401 0000 4a13 d239
                  24d5 0900

17:33:30.645497 localhost > localhost: icmp: echo reply
                  4500 0024 0023 0000 ff01 bdb3 7f00 0001
                  7f00 0001 0000 e1db d401 0000 4a13 d239
                  24d5 0900
```