1. Que font les deux programmes suivants ?
2. Décrivez en détail le code et son fonctionement.
3. Le code 'server.c' comporte une boucle infinie. Comment pourrait-on l'interrompre proprement ?
4. Le code du client devient problématique si la connecion réseau est lente ou mauvaise. Pourquoi ? Comment-y remédier ?

makefile

```
CC=gcc
CFLAGS=-Wall -g --pedantic

all: client server

client: client.o
	$(CC) $(CFLAGS) -o client client.o

client.o: client.c common.h
	$(CC) $(CFLAGS) -c -o client.o client.c

server: server.o
	$(CC) $(CFLAGS) -o server server.o

server.o: server.c common.h
	$(CC) $(CFLAGS) -c -o server.o server.c

clean:
	rm -rf *o client server
```

common.h

```
#ifndef COMMON_H
#define COMMON_H

#define die(issue) { perror(issue); exit(EXIT_FAILURE); }

#endif
```

server.c

```c
#include <stdio.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <netinet/in.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <time.h>

#include "common.h"
#define MAX_PENDING 5

void pa( struct sockaddr_in *address,  int port ) {
  memset(address, 0, sizeof(address));
  address->sin_family = AF_INET;
  address->sin_addr.s_addr = htonl(INADDR_ANY);
  address->sin_port = htons(port);
}

int ms( int port ) {
  struct sockaddr_in address;
  int sock = socket(PF_INET, SOCK_STREAM, 0);
  if( sock < 0 ) {
    die("Failed to create socket");
  }
  pa( &address, port );
  if( bind( sock,
            (struct sockaddr *) &address,
            sizeof(address)
            ) < 0  )
    {
      die("Failed to bind the server socket");
    }
  if (listen(sock, MAX_PENDING) < 0) {
    die("Failed to listen on server socket");
  }
  return sock;
}

void hc( int clientSock ) {
  time_t t;
  struct tm *now;
  time( &t );
  now = gmtime( &t );
  write( clientSock, now, sizeof(struct tm) );
  close( clientSock );
}

void run( int serverSock ) {
  while( 1 ) {
    struct sockaddr_in clientAddress;
    unsigned int clientLength = sizeof(clientAddress);
    int clientSock;
    printf( "Waiting for incoming connections\n");
    clientSock =
      accept(serverSock, (struct sockaddr *) &clientAddress, &
         clientLength );
    if( clientSock < 0 ) {
      die("Failed to accept client connection");
    }
    printf( "Client connected: %s\n", inet_ntoa(clientAddress.sin_addr))
       ;
    hc(clientSock);
  }
}

int main( int argc, char **argv ) {
  int servSock;
  int port;

  if (argc != 2) {
    fprintf(stderr, "USAGE: %s <port>\n", argv[0]);
    exit(EXIT_FAILURE);
  }

  port = atoi(argv[1]);

  servSock = ms( port );

  printf( "Server running on port %d'\n", port );

  run( servSock );

  close(servSock);

  return EXIT_SUCCESS;
}
```

client.c

```c
#include <stdio.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <netinet/in.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <time.h>

#include "common.h"

void pa( struct sockaddr_in *address, const char *host, int port ) {
  memset(address, 0, sizeof(struct sockaddr_in));
  address->sin_family = AF_INET;
  inet_pton( AF_INET, (char*) address, &(address->sin_addr) );
  address->sin_port = htons(port);
}

int ms( const char *host, int port ) {
  struct sockaddr_in address;
  int sock = socket(PF_INET, SOCK_STREAM, 0);
  if( sock < 0 ) {
    die("Failed to create socket");
  }
  pa( &address, host, port );
  if( connect(sock, (struct sockaddr *) &address, sizeof(struct
      sockaddr_in)) < 0) {
    die("Failed to connect with server");
  }
  return sock;
}

int get( int socket, struct tm *answer ) {
  int r = read( socket, answer, sizeof(struct tm) );
  return r;
}

void display( struct tm *t ) {
  printf( "%d/%d/%d - %d:%d:%d\n",
          t->tm_mday, (t->tm_mon+1), (t->tm_year+1900),
          t->tm_hour, t->tm_min, t->tm_sec );
}

int main(int argc, char *argv[]) {
  int sock;
  char *host;
  int port;
  struct tm answer;

  if (argc != 3) {
    fprintf(stderr, "USAGE: %s <host> <port>\n", argv[0]);
    exit(EXIT_FAILURE);
  }

  host = argv[1];
  port = atoi(argv[2]);

  sock = ms( host, port );

  if( get(sock,&answer) <  0 ) {
    die( "Reception error." );
  }

  close(sock);

  display(&answer);

  exit(EXIT_SUCCESS);
}
```