Systemnahe Software I
Allgemeine Informatik III (WS 97/98)

Abgabetermin 13. Januar 1998

14 Schiffeversenken (20 Punkte)

Schreiben Sie in Teams von bis zu drei Personen ein kleines Programm, welches Ihnen erlaubt, über das Netz Schiffeversenken zu spielen. Für die Kommunikation dürfen Sie mein Modul `sock` verwenden. Da verschiedene Versionen von Schiffeversenken gespielt werden, sollten folgende Regeln verwendet werden:

- Beide Seite dürfen immer zur gleichen Zeit einen Schuß abgeben. Dies ist unabhängig von der Anzahl der noch schwimmenden Schiffe. Auch Treffer geben keine Bonusschüsse.
- Gruppen von 2 Personen müssen nicht melden, wenn ein Schiff versenkt worden ist. Bei Dreiergruppen sollte dies geschehen.
- Nach jedem Schuß bekommen die Spieler das Ergebnis des Schusses angezeigt, sobald die Gegenseite es meldet (also Feuer oder Wasser).
- Es wird auf einem zehn auf zehn Felder großem Feld gespielt.
- Es gibt je ein Schiff der Länge zwei, vier und fünf, sowie zwei Schiffe der Länge drei.
- Alle Schiffe dürfen nicht diagonal gesetzt werden.

Als Ausgabe reicht es die zehn Zeilen mit `printf` auszugeben. Graphische Sonderleistungen sind nett, müssen aber nicht sein. Also einfach immer neuen Stand neu ausgeben.

Schreiben Sie mindestens zwei Module (zum Beispiel könnte die Ausgabe abgekapselt werden), und ein Makefile. Die Dateien `sock.h` und `sock.o` können von der Homepage der Vorlesung gezogen werden.

Information zu Sock

Da `sock` Sockets verwendet, müssen die Compileroptionen `-lsocket` und `-lnsl` verwendet werden. Dies koennte zum Beispiel so aussehen:

```
gcc -Wall -c yourfile.c
gcc -Wall -lsocket -lnsl -o yourfile yourfile.o sock.o
```

Es werden folgende Funktionen in `sock` zu Verfügung gestellt:

init_sock(port) Öffnet eine Verbindung auf Port *port*.

sendline(othersname, port, sock, tosend) Sendet bis zu 1024 Zeichen an *othersname*[1], der an Port *port* lauscht. *sock* ist der Filedescriptor, der mit `init_sock` erzeugt wurde.

receiveline(othersname, port, sock, res) Empfängte eine mit `sendline` verschickte Zeile.

close_sock(sock) Macht die Verbindung wieder zu.

Hier noch eine kleine Beispielanwendung:

```
#include "sock.h"

void main(int argc, char *argv[]) {
   int sock;
   char hostname[64];
   char rec[1024];
   int port;
   if (argc == 3) { /* 2 Kommandozeilenargumente - hostname und Port */
      strcpy(hostname, argv[1]);
      sock = init_sock(&port); /* init to send */
      printf("I am now on port %d.\n", port);
      port = atoi(argv[2]);
      printf("Sending %s to %s on port %d.\n", "Hallo arg 3", hostname, port);
      sendline(hostname, port, sock, "Hallo arg 3");
      printf("Line sent.\n");
   } else {
      sock = init_sock(&port); /* init to receive */
      printf("I am now on port %d.\n", port);
      if (argc == 2) {
         strcpy(hostname, argv[1]);
      } else {
         printf("Please enter other player's hostname: ");
         scanf("%s", hostname);
      }
      printf("Please enter other player's port: ");
      scanf("%d", &port);
      printf("Sending %s to %s on port %d.\n", "Hallo no args", hostname, port);
      sendline(hostname, port, sock, "Hallo no args");
   }
   receiveline(hostname, port, sock, rec); /* Blocks until line received */
   printf("I got: %s.\n", rec);
   printf("Sending %s to %s on port %d.\n", "Another line", hostname, port);
   sendline(hostname, port, sock, "Another line");
   receiveline(hostname, port, sock, rec);
   printf("I got: %s.\n", rec);
   close_sock(sock);
   exit(0);
}
```

[1] Zum Beispiel turing oder thales.