SYSPROG BEISPIEL 3

Aufgabenstellung

Um sich die Zeit während besonders langweiliger Vorlesungen bestmöglich zu vertreiben, beschließen Sie eine einfache Tic-Tac-Toe-Implementierung mit 'künstlicher Intelligenz' zu schreiben. Die Regeln des Spiels sind primitiv und bedürfen keiner weiteren Erklärung (diejenigen, die es wirklich nicht kennen, mögen einen Blick auf *http://de.wikipedia.org/wiki/Tic_Tac_Toe* werfen). Das Augenmerk sollte auf den Einsatz von Interprozesskommunikation, in dem Fall mittels Shared Memory und Synchronisation mittels Semaphoren, gelegt werden.
Es sind also zwei Programme, *Server* und *Client* zu programmieren.

Die gesamte Spiellogik (und der einfache CPU-Gegner) sollen dabei auf Seite des Serverprozesses implementiert werden - der Client bietet dem Benutzer nur ein praktisches Interface an und muss ledliglich das Setzen von Symbolen auf bereits belegten Felder verbieten; um die Gewinnchancen etwas zu erhöhen bietet er auch die Möglichkeit, gleich zwei Symbole auf einmal zu platzieren (wovon aber beide Plätze wieder frei sein müssen). Der Server sollte diesen Betrug natürlich feststellen und dies unmittelbar als verloren für den Client werten! Die Cheat-Erkennung kann auf Wunsch auch mit der Option `c` deaktiviert werden:

```
SYNOPSIS
    ttt-server [-c]
    ttt-client
```

Details

Server-Prozess

Die Aufgaben des Servers nochmal zusammengefasst:

- Shared Memory und Semaphoren anlegen (und am Ende wieder freigeben!)
- nach jedem Zug des Clients mittels einer lokalen Kopie des Spielfelds auf Cheating überprüfen (wenn dieser Modus nicht deaktiviert wurde!)
- selbst ein Symbol an einer zufälligen freien Position tätigen - jedes der freien Felder soll dabei mit der selben Wahrscheinlichkeit gewählt werden!
- auf Sieg/Verlust/Unentschieden prüfen und dies im Falle dem Client mitteilen

Der Server läuft in einer Endlosschleife und läßt sich lediglich per Ctrl+C (*SIGINT*) beenden. Achten Sie darauf dass in diesem Fall alle angelegten Ressourcen ordnungsgemäß wieder freigegeben werden!

Client-Prozess

Der Client soll beim Starten zunächst versuchen zum Server zu verbinden ('verbinden' heißt in dem Fall, einfach zu überprüfen ob die Semaphore die der Server angelegen sollte schon existieren - falls sie nicht existieren, wird mit einer aussagekräftigen Fehlermeldung abgebrochen). Bei Erfolg wird die aktuelle Spielposition sowie eine Begrüßungsnachricht ausgegeben, mit dem Hinweis dass mit dem Befehl `h` weitere Hilfe zur Steuerung angefordert werden kann.

Dem Spieler stehen folgende Befehle zur Verfügung:

- `s` - Spielfeld anzeigen (dies wird nach jedem Zug automatisch gemacht)
- `p <xy>` - Symbol platzieren auf Position xy ($x, y \in \{1, 2, 3\}$ - wobei (1,1) die Ecke links oben bezeichnet)
- `c <xy> <xy>` - zwei Symbole platzieren (Cheat-Modus)
- `n` - Neues Spiel starten
- `q` - Spiel beenden (Position bleibt weiterhin bestehen!)

Falls ein nicht existierender Befehl eingegeben wurde, sollte eine entsprechende Fehlermeldung ausgegeben werden und zusätzlich nochmal die Liste aller möglichen Kommandos angezeigt werden. Falls der Befehl existiert, aber falsch verwendet wurde (z.B. `p 45` oder `c 13`), sollte der Benutzer auch auf den Fehler aufmerksam gemacht werden (z.B. *Ërror: field coordinates out of bounds!"*).

Beim Empfang eines besonderen Ereignisses vom Server (Gewinn, Verlust, Verlust wegen Cheaten etc.) soll eine entsprechende Nachricht ausgegeben werden und nach einem Tastendruck fortgesetzt werden. Der erste Zug pro Spiel soll immer vom Client ausgeführt werden.

Hinweise

Um eine Kommunikation zwischen Server und Client abseits des Spielfelds zu ermöglichen (der Server muss den Client z.B. benachrichtigen falls er verloren oder gewonnen hat), ist es ratsam, außer dem Array noch eine Variable im Shared Memory zu definieren um sich mit Statusnachrichten auszutauschen.
Bitte beachten Sie auch die Allgemeinen Hinweise zu Beispielgruppe 3 und die Richtlinien für die Erstellung von C-Programmen.