ulm university universität
uulm

Institut für Numerische Mathematik

Dr. Andreas F. Borchert
Fabian Berstecher

29. Juni 2017
Blatt 9

Systemnahe Software II (SS 2017)

Abgabe bis zum 6. Juli 2017, 16:00 Uhr

Lernziele:

- Parallele Sitzungen auf Basis ereignisgesteuerter Behandler und dem Systemaufruf *poll*

Aufgabe 10: Multiplayer-Mastermind

Diese Aufgabe baut auf der Aufgabe 9 des letzten Übungsblattes auf. Das Ziel ist die Weiterentwicklung, die es beliebigen vielen Spielern erlaubt, gemeinsam an einer Mastermind-Partie teilzunehmen.

Die Implementierung des Dienstes muss auf Basis des *poll*-Systemaufrufs ereignisgesteuert in einem Prozess erfolgen. Dabei sind alle Ein- und Ausgaben über die Netzwerkverbindungen nicht blockierend abzuwickeln, d.h. das Warten auf neue Verbindungen, die Eingabe von den Klienten und die möglicherweise anstehenden Ausgaben sind alle in einem Aufruf von *poll* zu integrieren und dann sind die eingetretenen Ereignisse mit entsprechenden Behandlern abzuarbeiten. Sie können dazu aus der Vorlesungsbibliothek das Modul *mpx_session* verwenden. Es steht Ihnen aber auch frei, dies selbst vollständig auf der Basis von Systemaufrufen umzusetzen. Wenn Sie sich für *mpx_session* entscheiden, sollten Sie mit der in der Vorlesung vorgestellten Implementierung von *multiplexor* und *mpx_session* vertraut sein. Insbesondere relevant für das Verständnis sind die Funktionen *run_multiplexor* und *setup_polls* in *multiplexor.c*. Das Modul *multiplexor* verpackt dabei *poll* mit einer ereignisgesteuerten Abarbeitung und das darauf aufbauende Modul *mpx_session* fügt dann einzelne Pakete zusammen, um daraus mit Hilfe eines regulären Ausdrucks einzelne, in sich abgeschlossene Anfragen zu extrahieren und Behandlern auf der nächst höheren Ebene zur Verfügung zu stellen. Die hier wesentliche Funktion ist *mpx_input_handler* in *mpx_session.c*.

Der Server soll wie folgt gestartet werden können. Als Argumente werden der Port und der Dateiname angegeben.

```
thales$ mastermind localhost:33033
```

Als Klient können Sie *telnet* verwenden, hier ein beispielhafter Spielverlauf.

```
thales$ telnet localhost 33033
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
Welcome, please tell me your name (just letters).
Max
Max: joins us.
Franz: joins us.
3164
Max: Parameters: len = 4, colours = 6, unique = true
Max: 3164  Black: 1 White: 2
Franz: 5614  Black: 0 White: 3
5463
Max: 5463  Black: 1 White: 2
Franz: 5136  Black: 0 White: 4
4362
Max: 4362  Black: 2 White: 0
Franz: 1365  Black: 4 White: 0
Franz: won the game
quit
Bye!
Connection to localhost closed by foreign host.
thales$
```

Hier der gleiche Spielverlauf aus Sicht des anderen Mitspielers:

```
thales$ telnet localhost 33033
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
Welcome, please tell me your name (just letters).
Franz
Franz: joins us.
Max: Parameters: len = 4, colours = 6, unique = true
Max: 3164  Black: 1 White: 2
5614
Franz: 5614  Black: 0 White: 3
Max: 5463  Black: 1 White: 2
5136
Franz: 5136  Black: 0 White: 4
Max: 4362  Black: 2 White: 0
1365
Franz: 1365  Black: 4 White: 0
Franz: won the game
Max: leaves.
quit
Bye!
Connection to localhost closed by foreign host.
thales$
```

Wie Sie sehen, erfolgt die Kommunikation in diesem Beispiel in der Weise eines *chat*, d.h. alle beteiligten Seiten erhalten jederzeit unsychronisiert Nachrichten und können umgekehrt etwas versenden. Das einzige verbliebene strukturierende Element sind hier die Zeilentrenner CR LF. Prinzipiell könnte das weiter formalisiert werden, um die einzelnen Nachrichten und Statusreports formal leichter voneinander trennen zu können, aber dies müssen Sie nicht tun.

Hinweise:

Grundsätzlich empfiehlt es sich, die Abwicklung einer Sitzung und das Aufsetzen des Diensts in separate Module zu verpacken. Es steht Ihnen frei, die Vorlesungsbibliothek für Ihre Implementierung zu verwenden. Außerdem muss überprüft werden, ob der vom Spieler gewünschte Benutzername schon verwendet wird, in diesem Fall ist eine entsprechende Meldung auszugeben.

Ihre Lösung können Sie wiederum mit Hilfe von *tar* verpacken und dann mit *submit* einreichen:

```
tar cvf mastermind.tar *.[ch] [mM]akefile
submit ss2 10 team [notes] mastermind.tar
```

Viel Erfolg!