IPC-Übungsaufgaben

Franz Geischläger, Gerald Stoll, Josef Reisinger

Inhalt

1 Aufgabe – *Ausgabe von Files*

1.1 Angaben

Entwickeln Sie Client-Server-Programme welche folgende Funktionalität erfüllen:

- Entwicklung einer Client-Server Anwendung
- Der Client übergibt dem Server über eine Pipe die Namen der Dateien die ausgegeben werden sollen. Außerdem wird für jedes File eine Anzahl übergeben, wie oft die Datei ausgegeben werden soll. Die Übergabe der Parameter erfolgt im json-Format.
- Der Server schickt die Daten über eine Message Queue an den Client zurück, der diese entsprechend ausgibt.
- Zusätzlich übergibt der Server einen Log-Dämon-Prozess den Dateinamen und die Anzahl der Ausgaben. Der Log-Dämon speichert dann diese Daten zusammen mit Datum und Uhrzeit in einer Logdatei.

1.2 Abgaben

- Protokoll
    - Abgabe im pdf-Format
    - mit Aufgabenstellung
    - mit Blockschaltbild
    - mit Erklärungen wie die Aufgabenstellung gelöst wurde
    - mit dokumentierten Source-Code
    - Funktionsnachweise mit Screenshots, Consolenausgaben, … und Erklärungen die die Funktionsweise entsprechend nachweisen

- Files
    - Abgabe als zip-Archiv (.zip)
    - alle Quelldateien
    - alle ausführbaren Dateien
    - alle Dokumente und Beschreibungen

- Hochladen in MS-Teams

2 Aufgabe – *Filterprogramm*

2.1 Angaben

Entwickeln Sie Client-Server-Programme welche folgende Funktionalität erfüllen:

- Entwicklung einer Client-Server Anwendung
- Der Client übergibt dem Server über eine Pipe den Inhalt eines Files, welches die Wörter "Schule", "FSST" und "IPC" enthält.
- Der Server filtert die Namen "Schule" gegen "Freizeit", "FSST" gegen "Sport" und "IPC" gegen "WhatsApp" und schickt die Daten an den Client über eine FIFO zurück, der diese danach entsprechend ausgibt.
- Zusätzlich übergibt der Server im json-Format einen Log-Dämon-Prozess die Namen der gefilterten Zeichenketten mit der Anzahl der Vorkommnisse im File. Der Log-Dämon speichert dann diese Daten zusammen mit Datum und Uhrzeit in einer Logdatei.

2.2 Abgaben

- Protokoll
  - Abgabe im pdf-Format
  - mit Aufgabenstellung
  - mit Blockschaltbild
  - mit Erklärungen wie die Aufgabenstellung gelöst wurde
  - mit dokmentierten Source-Code
  - Funktionsnachweise mit Screenshots, Consolenausgaben, … und Erklärungen die die Funktionsweise entsprechend nachweisen

- Files
  - Abgabe als zip-Archiv (.zip)
  - alle Quelldateien
  - alle ausführbaren Dateien
  - alle Dokumente und Beschreibungen

- Hochladen in MS-Teams

3 Aufgabe – *Matrizenmultiplikation*

3.1 Angaben

Entwickeln Sie Client-Server-Programme welche folgende Funktionalität erfüllen:

- Entwicklung einer Client-Server Anwendung
- Der Client liest aus einer geeigenten Datei die Werte für die einzelnen Matrizen ein und übergibt die Werte im json Format an den Server.
- Der Server berechnet das Ergebnis der Matrizenmultiplikation und übergibt die Werte über eine Message-Queue im json-Format dem Client zurück, der diese danach entsprechend ausgibt.
- Zusätzlich übergibt der Server über eine FIFO die Größe der Ergebnismatrix an einen Log-Dämon-Prozess. Der Log-Dämon speichert dann diese Daten zusammen mit Datum und Uhrzeit in einer Logdatei.

3.2 Abgaben

- Protokoll
  - Abgabe im pdf-Format
  - mit Aufgabenstellung
  - mit Blockschaltbild
  - mit Erklärungen wie die Aufgabenstellung gelöst wurde
  - mit dokmentierten Source-Code
  - Funktionsnachweise mit Screenshots, Consolenausgaben, … und Erklärungen die die Funktionsweise entsprechend nachweisen

- Files
  - Abgabe als zip-Archiv (.zip)
  - alle Quelldateien
  - alle ausführbaren Dateien
  - alle Dokumente und Beschreibungen

- Hochladen in MS-Teams

4 Aufgabe – *Determinatenberechnung*

4.1 Angaben

Entwickeln Sie Client-Server-Programme welche folgende Funktionalität erfüllen:

- Entwicklung einer Client-Server Anwendung
- Der Client liest aus einer geeigenten Datei die Werte für die einzelnen Matrizen ein und übergibt die Werte im json Format an den Server.
- Der Server berechnet die Determinate der Matrix und gibt das Ergebnis über ein UNIX-Socket im json-Format dem Client zurück, der diese danach entsprechend ausgibt.
- Zusätzlich übergibt der Server über eine FIFO das Ergebnis der Determinate an einen Log-Dämon-Prozess. Der Log-Dämon speichert dann diese Daten zusammen mit Datum und Uhrzeit in einer Logdatei.

4.2 Abgaben

- Protokoll
  - Abgabe im pdf-Format
  - mit Aufgabenstellung
  - mit Blockschaltbild
  - mit Erklärungen wie die Aufgabenstellung gelöst wurde
  - mit dokmentierten Source-Code
  - Funktionsnachweise mit Screenshots, Consolenausgaben, … und Erklärungen die die Funktionsweise entsprechend nachweisen

- Files
  - Abgabe als zip-Archiv (.zip)
  - alle Quelldateien
  - alle ausführbaren Dateien
  - alle Dokumente und Beschreibungen

- Hochladen in MS-Teams

5 Aufgabe – *Bauer Sepp*

5.1 Angaben

Entwickeln Sie Client-Server-Programme welche folgende Funktionalität erfüllen:

- Entwicklung einer Client-Server Anwendung
- Allgemeines
  Es ist März und der Acker von Bauer Sepp muss gepflügt werden. Sepp gibt dazu seinem Sohn Bruno jeweils am Morgen den Auftrag eine Teilfläche des Ackers zu pflügen (in Anzahl m2). Für seine Buchführung benötigt Sepp jeden Abend die Kosten für das Pflügen des entsprechenden Ackerteils in Franken. Bruno berechnet die Kosten wie folgt: Grundkosten 35.- plus 1.- pro gepflügtem m2. Dann übermittelt er den Betrag an Sepp, der die aufgelaufenen Gesamtkosten berechnet. Es gibt allerdings auch Tage an denen Bruno nicht pflügen muss, z.B. wenn das Wetter zu schlecht ist, in diesem Fall setzt Bauer Sepp die Anzahl zu Pflügender m2 auf 0.
- Technischer Hintergrund
  Bauer Sepp und sein Sohn Bruno werden durch zwei verwandte Prozesse realisiert (Vater und Sohn). Die Kommunikation zwischen Sepp und Bruno geschieht über Message Queues. Bauer Sepp übergibt seinem Sohn am morgen jeweils eine Meldung mit der Anzahl zu pflügender m2. Wenn Bruno nicht pflügen muss, sendet der Vater eine Null. Bruno bestätigt den Auftrag mit einer entsprechenden Meldung und beginnt die geforderte Anzahl m2 zu pflügen. Nach getaner Arbeit berechnet Bruno die Tageskosten und übermittelt den Betrag an Bauer Sepp, bei schlechtem Wetter einen Betrag von 0.-.

5.2 Abgaben

- Protokoll
  - Abgabe im pdf-Format
  - mit Aufgabenstellung
  - mit Blockschaltbild
  - mit Erklärungen wie die Aufgabenstellung gelöst wurde
  - mit dokmentierten Source-Code
  - Funktionsnachweise mit Screenshots, Consolenausgaben, … und Erklärungen die die Funktionsweise entsprechend nachweisen
- Files
  - Abgabe als zip-Archiv (.zip)
  - alle Quelldateien
  - alle ausführbaren Dateien
  - alle Dokumente und Beschreibungen
- Hochladen in MS-Teams

6 Aufgabe – *Bauernhof-Most*

6.1 Angaben

Entwickeln Sie Client-Server-Programme welche folgende Funktionalität erfüllen:

- Entwicklung einer Client-Server Anwendung
- Allgemeines
  Es ist Herbst und Bauer Sepp möchte sein Fallobst zu Most verarbeiten. Einige Nachbarskinder und seine eigenen Kinder sammeln Äpfel und bringen sie zu ihm in die Scheune. Sobald die Kinder 100kg Äpfel gesammelt haben, kann Bauer Sepp mit dem Mosten beginnen.
- Technischer Hintergrund
  In diesem Beispiel stellt der Bauer Sepp und seine Mostmaschine einen Server dar (Apfelverwertung). Die Kinder entsprechen Clients, die einen Dienst in Anspruch nehmen (Abnahme der Äpfel). Server und Clients sind als selbständige und unabhängige Programme implementiert. Die Abgabe der Äpfel repräsentiert die Kommunikation von Client zu Server, die Antwort von Bauer Sepp entspricht der Kommunikation vom Server zum Client. Der Server (Bauer Sepp) terminiert, wenn er von den Kindern mehr als 100kg Äpfel erhalten hat. Pro Aufruf bringen die Kinder zwischen 5kg und 10kg Äpfel (Zufallsgenerator). Die Ausgabe des Servers sieht wie folgt aus (der Name des Clients wird ausgegeben):

  ```
  Bauer Sepp: heute ist Mosttag ...sammelt bitte Aepfel

  Bauer Sepp: habe 6 kg Aepfel von diva erhalten
  Bauer Sepp: jetzt sind es insgesamt 6 kg Aepfel

  Bauer Sepp: habe 8 kg Aepfel von diva erhalten
  Bauer Sepp: jetzt sind es insgesamt 14 kg Aepfel

  ....
  ```

  Pro Aufruf gibt der "Kinderprozess" folgende Meldung aus (Aufforderung mehr Äpfel zu sammeln):
  ```
  Kinder: Vater Sepp benoetigt noch 94 kg Aepfel bis er mosten kann
  ```

6.2 Abgaben

- Protokoll
  - Abgabe im pdf-Format
  - mit Aufgabenstellung
  - mit Blockschaltbild
  - mit Erklärungen wie die Aufgabenstellung gelöst wurde
  - mit dokmentierten Source-Code
  - Funktionsnachweise mit Screenshots, Consolenausgaben, … und Erklärungen die die Funktionsweise entsprechend nachweisen

- Files
    - Abgabe als zip-Archiv (.zip)
    - alle Quelldateien
    - alle ausführbaren Dateien
    - alle Dokumente und Beschreibungen

- Hochladen in MS-Teams

7 Aufgabe – *cat-grep-awk*

7.1 Angaben

Entwickeln Sie Client-Server-Programme welche folgende Funktionalität erfüllen:

- Entwicklung einer Client-Server Anwendung
- Allgemeines
  Realisieren Sie das Shellkommando
  cat inputfile | grep ´running´ | awk –F: ´{print $3}´
  mit Hilfe von Unix-Systemcalls und den Interprozessmechanismen **Pipe** und **Message Queues**.
- Input
  Der Inhalt des inputfile könnte folgendermaßen strukturiert sein:
  001:running:olaf:null
  002:blocked:sascha:null
  003:running:jens:null
  …

7.2 Abgaben

- Protokoll
  - Abgabe im pdf-Format
  - mit Aufgabenstellung
  - mit Blockschaltbild
  - mit Erklärungen wie die Aufgabenstellung gelöst wurde
  - mit dokmentierten Source-Code
  - Funktionsnachweise mit Screenshots, Consolenausgaben, … und Erklärungen die die Funktionsweise entsprechend nachweisen

- Files
  - Abgabe als zip-Archiv (.zip)
  - alle Quelldateien
  - alle ausführbaren Dateien
  - alle Dokumente und Beschreibungen

- Hochladen in MS-Teams

8 Aufgabe – *Vektor-Multiplikation1*

8.1 Angaben

Entwickeln Sie Client-Server-Programme welche folgende Funktionalität erfüllen:

- Entwicklung einer Client-Server Anwendung
- Der Client liest aus einer geeigenten Datei die Werte für die einzelnen Vektoren ein und übergibt die Werte im json Format an den Server.
- Der Server berechnet das Ergebnis der Vektormultiplikation (dyadisches Produkt) und übergibt die Werte über eine Message-Queue dem Client zurück, der diese danach entsprechend ausgibt.
- Zusätzlich übergibt der Server über eine FIFO die Größe der Ergebnismatrix an einen Log-Dämon-Prozess. Der Log-Dämon speichert dann diese Daten zusammen mit Datum und Uhrzeit in einer Logdatei.

8.2 Abgaben

- Protokoll
    - Abgabe im pdf-Format
    - mit Aufgabenstellung
    - mit Blockschaltbild
    - mit Erklärungen wie die Aufgabenstellung gelöst wurde
    - mit dokmentierten Source-Code
    - Funktionsnachweise mit Screenshots, Consolenausgaben, … und Erklärungen die die Funktionsweise entsprechend nachweisen

- Files
    - Abgabe als zip-Archiv (.zip)
    - alle Quelldateien
    - alle ausführbaren Dateien
    - alle Dokumente und Beschreibungen

- Hochladen in MS-Teams

9 Aufgabe – *Vektor-Multiplikation2*

9.1 Angaben

Entwickeln Sie Client-Server-Programme welche folgende Funktionalität erfüllen:

- Entwicklung einer Client-Server Anwendung
- Der Client liest aus einer geeigenten Datei die Werte für die einzelnen Vektoren ein und übergibt die Werte im json Format an den Server.
- Der Server berechnet das Ergebnis der Vektormultiplikation und übergibt die Werte über ein UNIX-Socket dem Client zurück, der diese danach entsprechend ausgibt.

9.2 Abgaben

- Protokoll
  - Abgabe im pdf-Format
  - mit Aufgabenstellung
  - mit Blockschaltbild
  - mit Erklärungen wie die Aufgabenstellung gelöst wurde
  - mit dokmentierten Source-Code
  - Funktionsnachweise mit Screenshots, Consolenausgaben, … und Erklärungen die die Funktionsweise entsprechend nachweisen

- Files
  - Abgabe als zip-Archiv (.zip)
  - alle Quelldateien
  - alle ausführbaren Dateien
  - alle Dokumente und Beschreibungen

- Hochladen in MS-Teams

10 Aufgabe – *cat-sort*

10.1 Angaben

Entwickeln Sie Client-Server-Programme welche folgende Funktionalität erfüllen:

- Entwicklung einer Client-Server Anwendung
- Der Client liest den Inhalt einer unsortierten Datei und übergibt den Inhalt an einen Serverprozess mittels einer FIFO IPC.
- Der Server sortiert den übergebenen Inhalt und sendet das Ergebnis an den Client über eine Message Queue zurück.
- Zusätzlich übergibt der Server die Anzahl der gelesen Zeilen und einen Zeitstempel im json-Format an einen Log-Dämon-Prozess. Der Log-Dämon speichert dann diese Daten in einer Logdatei.

10.2 Abgaben

- Protokoll
    - Abgabe im pdf-Format
    - mit Aufgabenstellung
    - mit Blockschaltbild
    - mit Erklärungen wie die Aufgabenstellung gelöst wurde
    - mit dokmentierten Source-Code
    - Funktionsnachweise mit Screenshots, Consolenausgaben, … und Erklärungen die die Funktionsweise entsprechend nachweisen

- Files
    - Abgabe als zip-Archiv (.zip)
    - alle Quelldateien
    - alle ausführbaren Dateien
    - alle Dokumente und Beschreibungen

- Hochladen in MS-Teams

11 Aufgabe – *Chat-Programm*

11.1 Angaben

Entwickeln Sie Client-Server-Programme welche folgende Funktionalität erfüllen:

- Entwicklung Sie ein chat-Programm mit Message Queue IPC
  - Two instances of the same program (chatwithme) are talking to each other over a UNIX message queue. They are peer-to-peer messaging
  - It takes following optional arguments:
    Usage: ***chatwithme <-k key> <-m my_msg_type> <-r rcvr_msg_type> <-u user_name>***
  - You must use getopt() function to handle the command line options

- How to chat?
  - Chatting between two users running from the same server
  - Messages should be displayed asynchronously, which means each user's message should be displayed immediately without being blocked or delayed.
  - Display the user name or "Other" for the other party before the message entered/received. See below for a sample output.
  - You may consider running two processes for each user, one for sending your own message and the other for receiving messages from the other end.

- How to stop the processes?
  - From the terminal, if a user enters a "bye" or "quit" message, the both processes stop their execution.
  - When the messaging session ends, all the processes involved should be terminated and the message queue and it's data should be removed by the kernel.

- Output example
  - User1
    $ chatwithme –k 12345 –m 5 –r 9 –u John
    John>>
    Other>>Hi John!
    John>>Hey Mary!
    John>>What's up?
    John>>
    Other>>Nothing much. A quick question for you…
    John>>Sure go ahead.
    John>>
    Other>>How difficult is it to learn Unix?
    John>>
    Other>>Is it hard?
    John>>Well…
    John>>Depending on.
    John>>bye

- User2
  $ chatwithme –k 12345 –m 9 –r 5 –u Mary
  Mary>>Hi John!
  Mary>>
  Other>>Hey Mary!
  Mary>>
  Other>>What's up?
  Mary>>Nothing much. A quick question for you…
  Mary>>
  Other>>Sure go ahead.
  Mary>>How difficult is it to learn Unix?
  Mary>>Is it hard?
  Mary>>
  Other>>Well…
  Mary>>
  Other>>Depending on.
  Mary>>bye
  can't send message

11.2 Abgaben

- Protokoll
  - Abgabe im pdf-Format
  - mit Aufgabenstellung
  - mit Blockschaltbild
  - mit Erklärungen wie die Aufgabenstellung gelöst wurde
  - mit dokmentierten Source-Code
  - Funktionsnachweise mit Screenshots, Consolenausgaben, … und Erklärungen die die Funktionsweise entsprechend nachweisen

- Files
  - Abgabe als zip-Archiv (.zip)
  - alle Quelldateien
  - alle ausführbaren Dateien
  - alle Dokumente und Beschreibungen

- Hochladen in MS-Teams