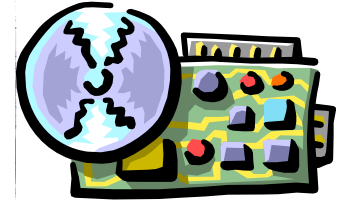




TI III: Operating and Communication Systems



WS 2007/08
Übungsblatt Nr. 5

Georg Wittenburg, M.Sc., AG Technische Informatik, Freie Universität Berlin

Ausgabe am 14.12.2007 — Abgabe spätestens 11.1.2008, 10:00 Uhr

Bitte bei der Abgabe beide Namen/Matr.Nr. der Mitglieder einer Gruppe, NUMMER DER ÜBUNG/TEILAUFGABE und DATUM auf den Lösungsblättern **nicht vergessen!** Darauf achten, dass die Lösungen beim richtigen Tutor/der richtigen Tutorin abgegeben werden.
Achten Sie bei Programmieraufgaben außerdem darauf, dass diese im Linuxpool kompilierbar sind.
Zu spät abgegebene Lösungen werden nicht mehr berücksichtigt!

1. Aufgabe: Begriffe (6 Punkte)

Beschreiben Sie jeden der folgenden Begriffe durch maximal zwei Sätze:
Bit, Signal, Multiplexing, Framing, Header, Piggybacking

2. Aufgabe: Kodierungsverfahren (8 Punkte)

- Berechnen Sie die Frame Check Sequence des Generatorpolynoms x^2+1 für die Bitfolge 1111001. Fügen Sie diese an die Bitfolge an und kodieren sie die resultierende Bitfolge mit der Manchesterkodierung. Vervollständigen Sie die Bitfolge mit durch Framing mit dem Frame 111. Ist Bitstuffing nötig um die Einzigartigkeit des Frames zu gewährleisten?
- Die CRC-gesicherte Bitfolge 1001110001010110 wurde empfangen. Ist bei der Übertragung ein Fehler aufgetreten, wenn das Generatorpolynoms x^2+1 verwendet wurde?

3. Aufgabe: Datenrate (2 Punkte)

Ein Kanal habe eine Bandbreite von 5 MHz. Wie viele Bit/s können gesendet werden, wenn digitale Signale mit 6 Levels verwendet werden (nehmen sie einen rauschlosen Kanal an)?

4. Aufgabe: Fifty/Fifty (6 Punkte)

Entscheiden Sie, ob folgenden Aussagen zutreffend oder nicht zutreffend sind, und begründen Sie Ihre Entscheidung:

- UDP Pakete werden aufgrund des nicht benötigten Verbindungsaufbaus bevorzugt für Multimedia-Anwendungen verwendet.
- Nach dem Theorem von Nyquist können unbegrenzte Datenraten in der Praxis erreicht werden.
- Um die maximal mögliche Datenrate zu ermitteln, muß man das Maximum der nach den Formel von Nyquist und Shannon berechneten Werte nehmen.

5. Aufgabe: Scheduler (12 Punkte)

Betrachten Sie die Implementierung des Schedulers des Linux-Kernels 2.6.18 (im Quelltext zum Download erhältlich unter <http://www.kernel.org/pub/linux/kernel/v2.6/linux-2.6.18.1.tar.bz2>), konkret die Dateien `kernel/sched.c` und `include/linux/sched.h`. Zusätzliche Dokumentation ist auf der Homepage der Veranstaltung verlinkt.

Erstellen Sie analog zu den in der Vorlesung zum Thema „Prozesse“ besprochenen Diagrammen ein Prozesszustandsdiagramm für diese konkrete Implementierung eines Schedulers. Vernachlässigen Sie hierbei die für das Auslagern von Prozessen verwendeten Zustände. Geben Sie stattdessen an, wo im Quelltext (Funktionsname, Zeilennummer) welche Zustandsübergänge implementiert sind.

Erstellen Sie wiederum analog zu den in der Vorlesung zum Thema „Scheduling“ besprochenen Diagrammen ein weiteres Diagramm, das die verwendeten Warteschlangen und sonstigen Datenstrukturen des Schedulers darstellt. Verwenden Sie zur Beschriftung sowohl die entsprechenden Namen der Variablen aus dem Quelltext als auch kurze Kommentare zur Funktion innerhalb des Schedulers.

6. Aufgabe: Webserver (16 Punkte)

Implementieren Sie in C einen einfachen Webserver. Das Programm soll als Kommandozeilenparameter eine Portnummer und ein Arbeitsverzeichnis erhalten und auf dem angegebenen Port auf einfache HTTP GET-Anfragen warten und diese korrekt bearbeiten (vgl. Folie 7.29). Der Arbeitsverzeichnisparameter soll angeben, an welcher Stelle im lokalen Dateisystem sich die HTML Seiten befinden. Geben Sie geeignete Ausgaben über die Abläufe innerhalb Ihres Webserver auf `stdout` aus, um die Funktionsweise zu verdeutlichen.

Schlagen Sie zu etwaigen Detailfragen im RFC 2616 (<http://www.w3.org/Protocols/rfc2616/rfc2616.html>) nach. Testen Sie Ihren Webserver durch geeignete Aufrufe durch einen gängigen Webbrowser, z.B. der URL `http://localhost:8080/index.html`, falls Ihr Webserver auf Port 8080 auf Verbindungen wartet.