CS 536 (Park) **Assignment IV** **Due: Sep. 26 (Fri.), 1997**

PROBLEM 1

(a) Read "Ethernet: Distributed packet switching for local computer networks" by R. Metcalf and D. Boggs. *Communications of the ACM*, 19(7):395–403, 1976. Give a 1-page summary/critique of the paper.

(b) Give a 3-page summary overview of the following aspects of UNIX system programming: signals (`sigaction`, `signal`, `kill`, `pause`) and interprocess communication (`pipe`, `mkfifo`, `semget`, `shmctl`). Organize your description around the main system calls (a subset is indicated in parenthesis). Make sure to explain the notions *atomicity*, *blocking/non-blocking*, *synchronous/asynchronous I/O*, *mutual exclusion and semaphores*, and *interrupts*, and in which system calls they play a role.

PROBLEM 2

(a) You will find a client/server application under ~park/pub/cs536; the client program is `client.c` and the server program is `server.c`. Reverse-engineer the code and explain, in a high-level way, what the server and client are doing. Then, give a line-by-line explanation of the code—you may group several lines into one unit, as you see fit, and give a modular description. The important criterion is that your answer makes clear that you understand all aspects of the code. If you find inefficiencies, indicate what and why, and suggest possible changes. Include in your answer a clear explanation of what trick is being used to direct the output of the server back to the client and why this is nontrivial.

(b) Extend the client/server application such that any UNIX command (you may ignore argument options) can be requested by the client to be executed on the server and its output returned to the client. The request format should be of the form

process-ID # command #

where '#' is a delimiter symbol. Your client program is to take as its command-line argument the name of the UNIX command to be executed. Test your program by first running the server application in the background, then executing four copies of the client in the background "simultaneously" with arguments `finger`, `ps`, `date`, and `hostname`, respectively.

Use the UNIX `script` facility to record the run-time session. Hand in the output of the script as well as the client and server code. You must provide adequate documentation in your code.

(c) Describe precisely how you would extend the server/client code to incorporate command-line arguments. Your description may need code segments to be concise in its presentation.

PROBLEM 3

(a) In the approximate analysis of Ethernet's CSMA/CD protocol shown in class, what simplifying assumptions were made to keep the analysis tractable? In what direction (if any) will the conclusion change (i.e., strengthen or weaken) if the simplying assumption(s) are relaxed?

(b) Consider a CSMA/CD protocol which uses *subexponential* backoff (e.g., linear, quadratic, etc.) instead of the exponential backoff mechanism discussed in class. Under what conditions is a subexponential scheme preferable to the exponential one? Under what conditions is the opposite true? What subexponential scheme (give the functional form) would you choose as a possible alternative to the exponential scheme? Why this particular functional form? Justify.

PROBLEM 4

Problem 3.12, 3.14 from P & D.