```
    .
    .
#include <sys/wait.h>   //new include

static const int MAXPENDING = 5; // Maximum outstanding connection requests

int main(int argc, char *argv[]) {
    time_t  ticks;  //variable to hold date and time data
    char sendbuffer[BUFSIZE]; // Buffer for sending data to the client
    unsigned int childProcCount = 0; // Number of child processes
    .
    .
    for (;;) { // Infinite for loop; runs forever

    // Wait for a client to connect
    int clntSock = accept(servSock, (struct sockaddr *) NULL, NULL);
    if (clntSock < 0)
      DieWithSystemMessage("accept() failed");

    pid_t processID = fork();
    if (processID < 0)
      DieWithSystemMessage("fork() failed");
    else if (processID == 0)
    { // If this is the child process

    close(servSock);         // Child closes parent socket

    // clntSock is connected to a client!
    snprintf(sendbuffer, sizeof(sendbuffer), "%.24s\r\n", ctime(&ticks)); //Create data and time string in outgoing buffer
    ssize_t numBytesSent = send(clntSock, sendbuffer, strlen(sendbuffer), 0); //Send date and time string to the client
    if (numBytesSent < 0)
      DieWithSystemMessage("send() failed");

      exit(0);                  // Child process terminates
    } // end child process code


    printf("with child process: %d\n", processID);
    close(clntSock);  // Parent closes child socket descriptor
    childProcCount++; // Increment number of child processes

    while (childProcCount)
    { // Clean up all zombies
    processID = waitpid((pid_t) - 1, NULL, WNOHANG); // Non-blocking wait
    if (processID < 0) // waitpid() error?
       DieWithSystemMessage("waitpid() failed");
    else if (processID == 0) // No zombie to wait on
       break;
    else
        childProcCount--; // Cleaned up after a child
    } //end zombie killing loop
  }  //end infinite for loop
  // NOT REACHED
}
```