Author:
Hacı Hasan Savan
1901042704

Design:
This is the design that we saw in lecture, but I added an extra fifo to handle easily.The first fifo takes command results and the other one is for sending command to server.

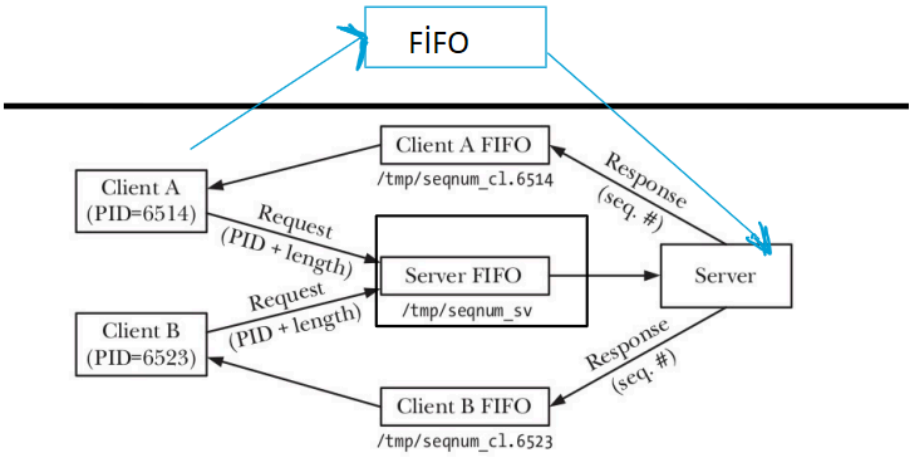


Figure 44-6: Using FIFOs in a single-server, multiple-client application

The file system server-client program is structured around a main process that oversees multiple subprocesses. This main process is responsible for managing client connections and allocating subprocesses to serve them. It acts as the parent process to all other subprocesses, handling tasks such as adding new clients to a queue and monitoring the number of active clients.

Client connection requests are received through a FIFO (named pipe) with the format "serverFifo". The main process places these requests into a waiting queue for processing. If a client requests to "connect", it is added to the queue. If a client requests to "tryconnect", the server checks if there is space available for the client. If there is no space, the client terminates.

Meanwhile, subprocesses (subservers) are continuously created and terminated as needed to serve individual clients. These subservers handle the actual communication and file operations with the clients.

System workflow is like that:

serverSide:	**clientSide**:
create serverFifo	.
create signal handlers	.
create semaphores	.
wait for connections	send connection req to server
	open clientFifo (cl_pid) to take responses
.	open clientFifoReq (cl_req_pid) to send command
	Send Command to server
Determine what is request	wait for the command result and write
Do the command's job	.
Send the result via cl_pid (in child)	.
go to "wait for connections"	go to "Send Command to server"

Synchronization:

To maintain synchronization during connections and file operations in the server-client system, named semaphores are employed. These semaphores ensure exclusive access to critical sections of code, preventing simultaneous access by multiple processes.

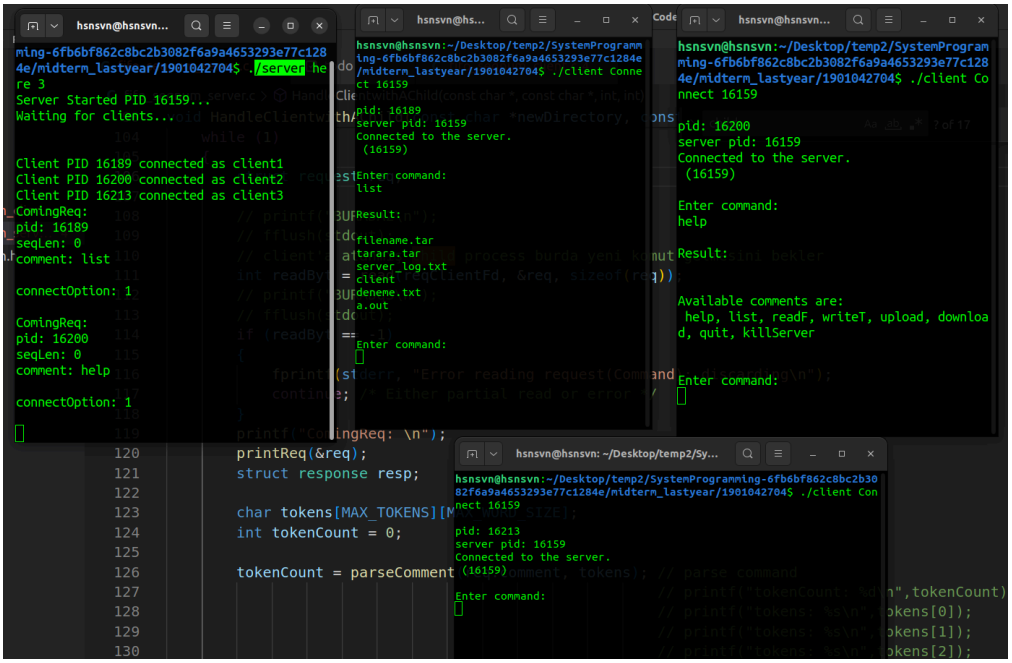
When connecting to the server, a semaphore called "`serverFifoSemaphore`" is utilized. This semaphore is initialized with a value of 1, indicating availability. Before writing a connection request to the server FIFO, the client waits on "`serverFifoSemaphore`" to lock it. Once acquired, the client proceeds with the write operation and releases the semaphore afterward.

Communication between clients and subservers occurs via blocking FIFOs, ensuring sequential message exchange and preventing conflicts. This synchronous communication model enhances system reliability by maintaining data integrity. On the client side we have no semaphore; they are all just on the server side.

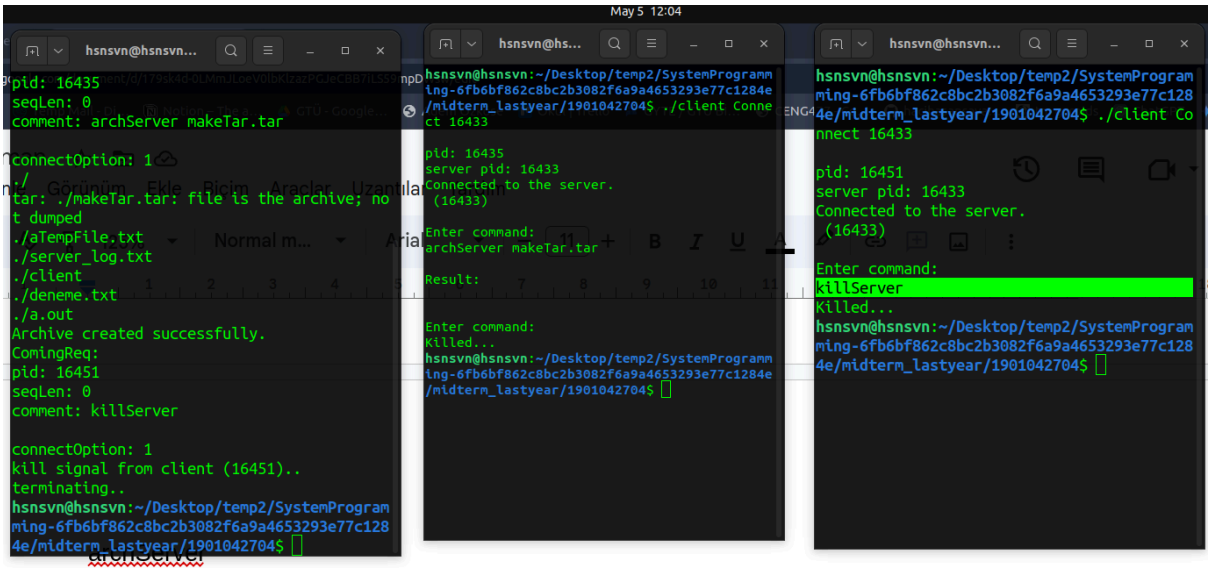
When adding logs, a named semaphore is used to synchronize access to the log file. Before writing, the function waits on the semaphore associated with the log file. After writing, the semaphore is released to allow other processes access.

During file operations, semaphores regulate access to critical code sections, ensuring only one process executes a particular operation at a time. Shared memory structures track file access and manage coordination between processes, enhancing system reliability and performance.

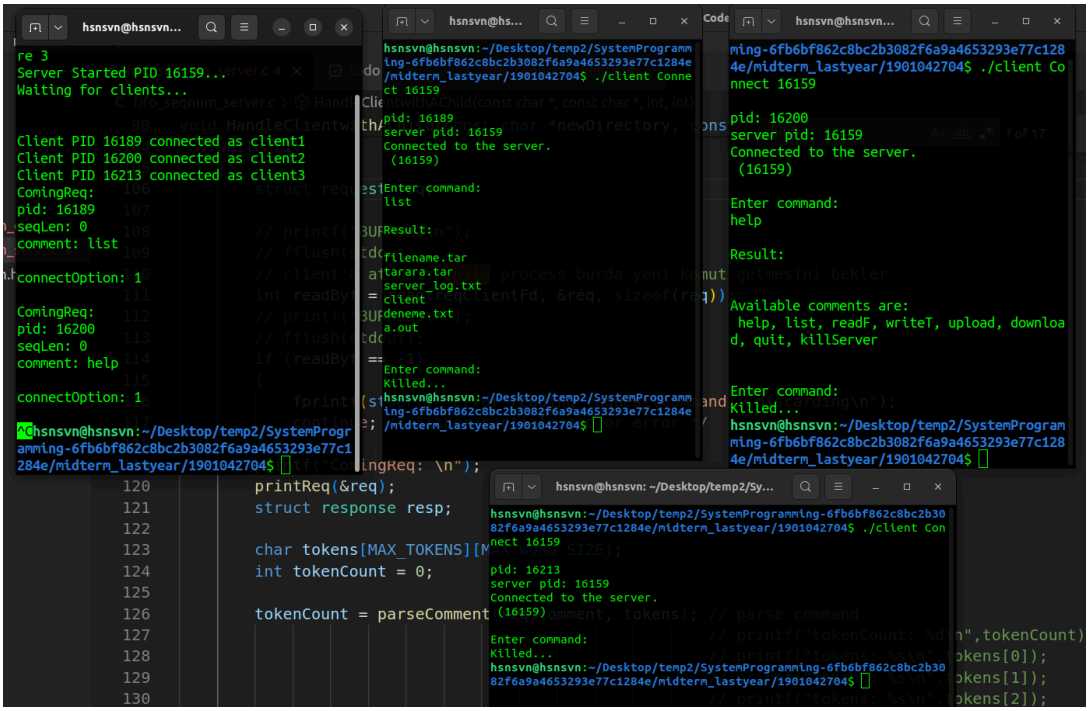
operations:



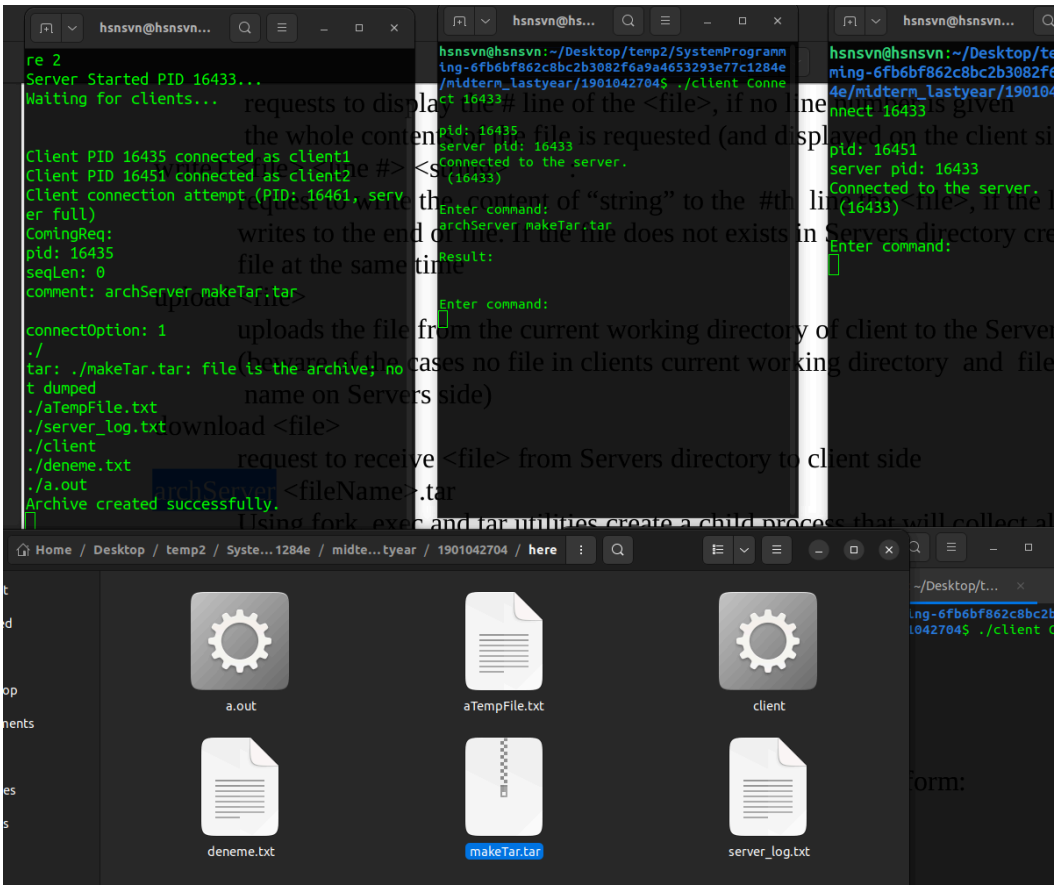
killServer

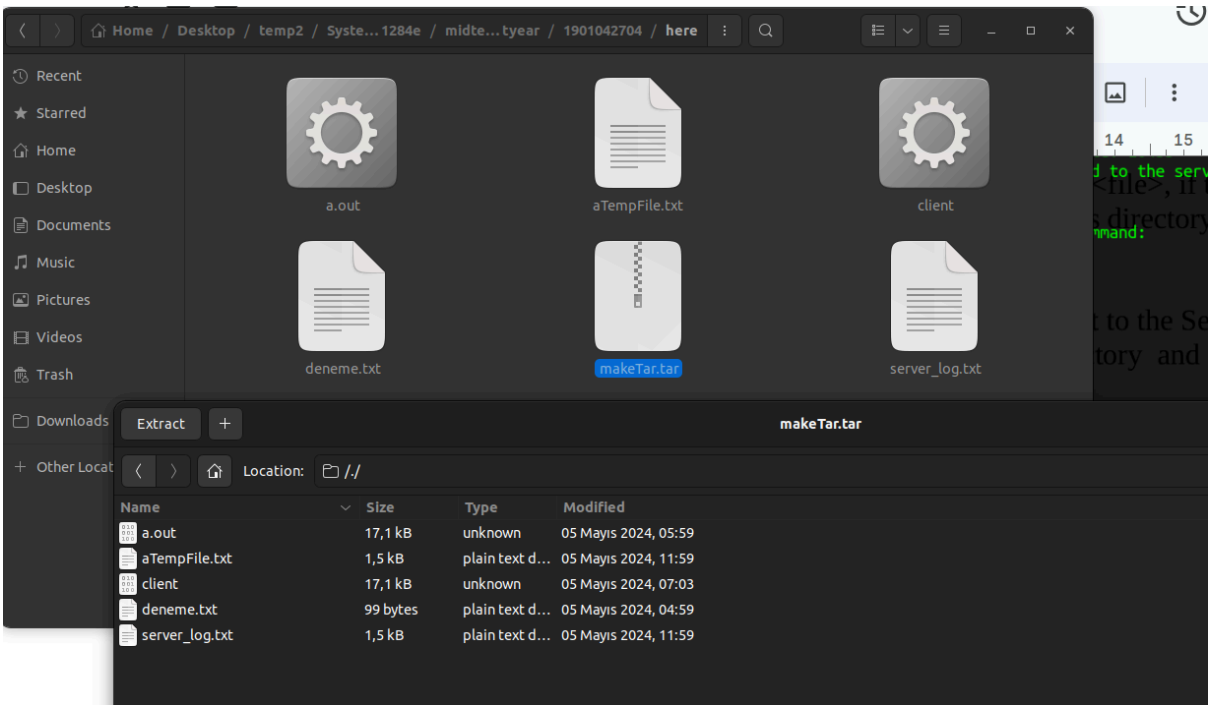


SIGINT

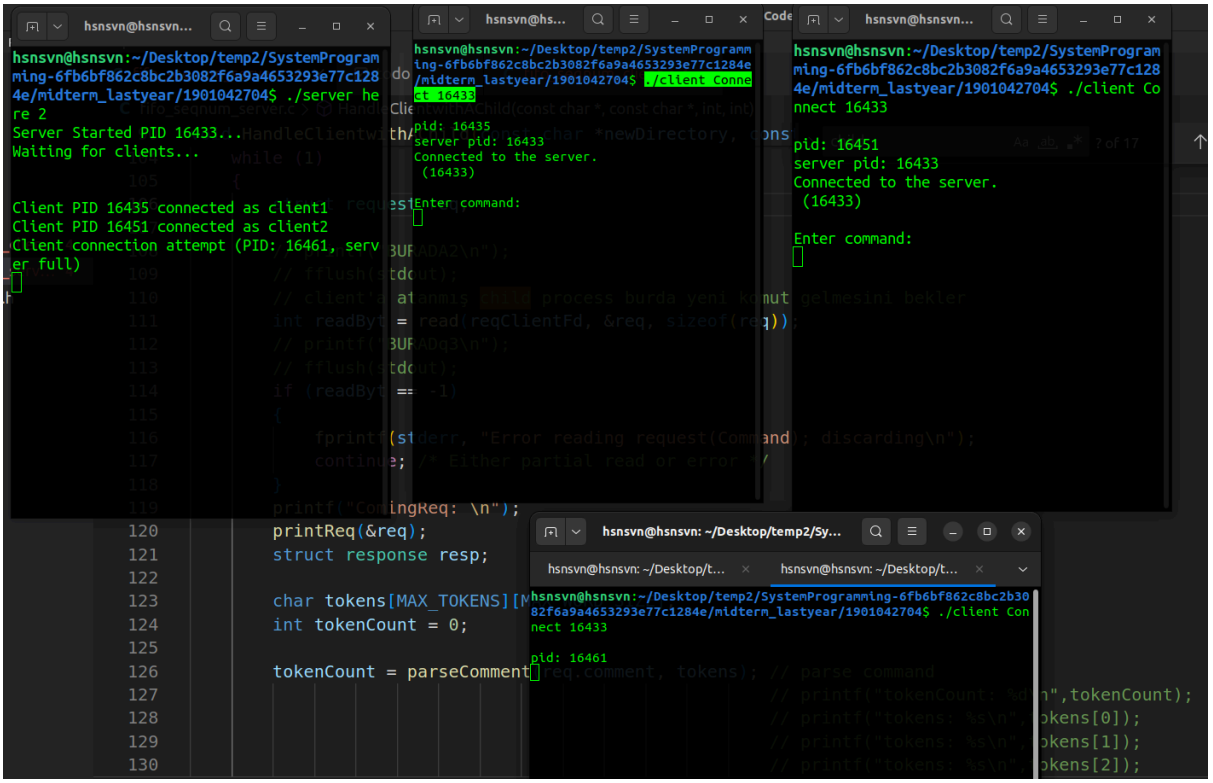

hsnsvn@hsnsvn...
re 3
Server Started PID 16159...
Waiting for clients...
Client PID 16189 connected as client1
Client PID 16200 connected as client2
Client PID 16213 connected as client3
ComingReq:
pid: 16189
seqLen: 0
comment: list
connectOption: 1
ComingReq:
pid: 16200
seqLen: 0
comment: help
connectOption: 1
hsnsvn@hsnsvn:~/Desktop/temp2/SystemProgramming-6fb6bf862c8bc2b3082f6a9a4653293e77c1284e/midterm_lastyear/1901042704$ ./client Connect 16159
pid: 16189
server pid: 16159
Connected to the server.
(16159)
Enter command:
list
Result:
filename.tar
tarara.tar
server_log.txt
client
deneme.txt
a.out
Enter command:
Killed...
hsnsvn@hsnsvn:~/Desktop/temp2/SystemProgramming-6fb6bf862c8bc2b3082f6a9a4653293e77c1284e/midterm_lastyear/1901042704$
pid: 16200
server pid: 16159
Connected to the server.
(16159)
Enter command:
help
Result:
Available comments are:
help, list, readF, writeT, upload, download, quit, killServer
Enter command:
Killed...
hsnsvn@hsnsvn:~/Desktop/temp2/SystemProgramming-6fb6bf862c8bc2b3082f6a9a4653293e77c1284e/midterm_lastyear/1901042704$
pid: 16213
server pid: 16159
Connected to the server.
(16159)
Enter command:
Killed...
hsnsvn@hsnsvn:~/Desktop/temp2/SystemProgramming-6fb6bf862c8bc2b3082f6a9a4653293e77c1284e/midterm_lastyear/1901042704$
printReq(&req);
struct response resp;
char tokens[MAX_TOKENS][
int tokenCount = 0;
tokenCount = parseComment

archServer

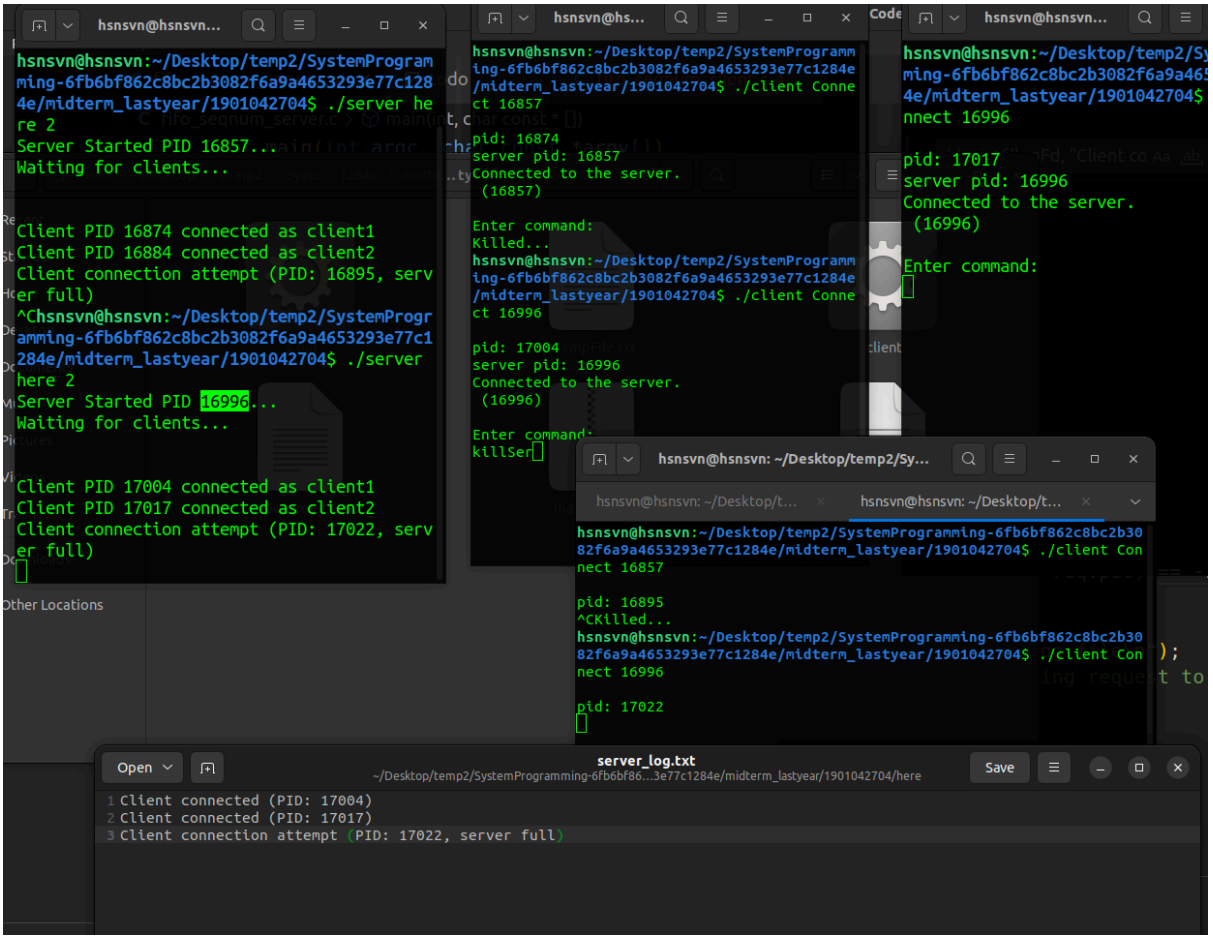

hsnsvn@hsnsvn...
re 2
Server Started PID 16433...
Waiting for clients...
Client PID 16435 connected as client1
Client PID 16451 connected as client2
Client connection attempt (PID: 16461, server full)
ComingReq:
pid: 16435
seqLen: 0
comment: archServer makeTar.tar
connectOption: 1
./
tar: ./makeTar.tar: file is the archive; not dumped
./aTempFile.txt
./server_log.txt
./client
./deneme.txt
./a.out
Archive created successfully.
hsnsvn@hsnsvn:~/Desktop/temp2/SystemProgramming-6fb6bf862c8bc2b3082f6a9a4653293e77c1284e/midterm_lastyear/1901042704$ ./client Connect 16433
pid: 16435
server pid: 16433
Connected to the server.
(16433)
Enter command:
archServer makeTar.tar
Result:
Enter command:
pid: 16451
server pid: 16433
Connected to the server.
(16433)
Enter command:
Home / Desktop / temp2 / Syste...1284e / midte...tyear / 1901042704 / here
a.out
aTempFile.txt
client
deneme.txt
makeTar.tar
server_log.txt



When server is full the newcomers will wait:



logfile:



it will also write the operations: