Read_cat.C

int read_file()

First, I ask the user to provide a file descriptor and number of bytes to be read. I then verify that the fd points to an open file (open for R or RW) and return myread(fd, buf, nbytes).

int myread(int fd, char *buf, int nbytes)

First, I allocate an openFile table entry and point it to the file to be read (running->fd[fd]) and set a MINODE to point to the open file (oftp->mptr). Next, I calculate the avil number of bytes to be read (filesize - current offset). Then, I run a while loop (while bytes still available & less than #bytes user supplied to read). Inside, I calculate the logical block number and starting byte to read from before converting lbk to a direct block using mailmans algorithm. I get the physical direct block into a buffer (readbuf) and set a char * pointer to the readbuf starting byte. I calculate remaining bytes in the block and then the max number of bytes to read in one operation (minimum of nbytes, avil, and remain). I then use memcpy to copy max bytes from readbuf into buf (parameter buffer) and decrement all counters by max. I increment count and offset by max, then continue back through the while loop until nbytes or avil is 0. Finally, I return fount (total number of bytes read in all).

int mycat(char *pathname)

First, I call open_file to open pathname for read and check it opened correctly. Next, I loop while myread is reading from the file descriptor and call puts(buf) to print the buffer to the screen. Finally, I call close_file(fd) to close the file I just opened after printing out its contents.