These are mostly two person projects

- Ethos coding projects

  1. **Remote file copy.** Copy recursively all files from current directory to destination directory, so that the destination is an exact copy of the current directory, including recursively subdirectories. If the file already exists at the destination, *it should not be copied.*

     Akash Edacheril Johny, Deepika Tripathi

  2. **Assignment submission.** Build a client/server system, which allows a client to submit an assignment up until some deadline. The system should check that the student is in the course, and that the assignment deadline has not yet expired.

     Francesco Marcantoni and Francesco Mantovani

  3. **Appointment calendar.** Write a client/server system in which there are appointment blocks, and users are allowed to schedule appointments at least 24 hours in advance during the schedule. At most one appointment can be scheduled for a future time.

     Sepideh Roghanchi and Muhammad Chaudhry

  4. **shell** Write a small shell for Ethos.

     Vincenzo Chiaramida and Francesco Pinci

  5. **ethosArchive dir** Given a directory as input, traverse all files recursively in the directory and write out them out as a sequence of values.

     Massimo Piras and Giovanni C. Monna

  6. **passwordEncryptedFile.** Given a password, use it to encrypt/decrypt a file in ethos. Your solution should carefully consider the security issues involved.

     Rohan Tadphale, Mrudula Borkar

  7. **packages** design a secure package system for Ethos. Describe the mechanisms and protections provided.

  8. **tiny database.** Consider a database consisting of slices of the empty interface. Write *select*, *join*, and *project*.

  9. **print server**. Write a client-server program to print files. (You can actually skip the printing part). This should enable files to be printed to a lab from outside of the lab, thus avoiding the firewall.

     Iacopo Olivo and Gabriele Milauro

  10. **fileSystemCheck**. Check that the Ethos file system contains all the necessary directories and files, and check that they are properly typed.

      Surekha Purushothaman

  11. **diskUsage.** (one person) Find the amount of space use by an Ethos file system

      Yumeng Yang

12. **sort.** Do an out-of-core sort, taking all the files in a directory and placing them in another directory by key. Files should be arbitrary type, using a generate sort package.

    Sriram Veturi, Suganya Sivakumar

13. **strings.** Print all the strings in an Ethos type. Must do all the types in kernel-Types.

    Saba Kathawala,Nishali D'Mello

14. **Assignment bidding.** Giving a set of assignments, and a maximum number of students per assignment, allow students to register for the desired assignment if available. This is a client/server system.

    Abhishek Ranjan

15. **Simple chat.** Client/server system. Post messages, and read messages from a shared chat system.

    Margaret Reid

16. **Authorization system.** Build a client-server system which will, for each directory, provide 2 sets of users, one for reading and one for writing. Clients should be able to request/submit files which will be granted by the server only if the user is listed as approved.

17. **Bell-LaPadula system.** Implement Bell-LaPadula in a client-server setting. Client programs should implement the *-property, while the Server should implement ss-property.

18. **Sandhu Separation of Duty.** Implement Sandhu's separation of duty in a client-server model.

19. **Room Scheduling** A room (such as a conference room) is available for scheduling. Members are allowed to schedule available 1 hour slots and free slots they have previously scheduled, the Secretary is allowed to both schedule and free anyones slots.

20. **Implement password authentication** Implement password authentication, you may assume any OS mechanism you need (Ethos may not implement currently all that you need) and describe the protections afforded.

- Go programs

    1. **Approvability.** Create a datastructure for approvability, and then create an approvability graph and enable steps to be taken against it.

    2. **MilSec.** Create a datastructure to represent a MilSec lattice with levels and compartments. Take a sequence of reads and writes, and show what happens on each transition.

- Research report. Read a few different related papers and do a report on them. You suggest the papers, they must be highly related to the course, and of course improved in advance.
- Analyze a system security mechanism. For example, IOS encryption security, trusted boot (TPM), hardware virtualization.
- Install a secure operating system (e.g., Qubes, Tails), and analyze its features for security.

  tails: Tommaso Massari