Stage	1	2	3	4	5	Sum
Points	3	7	4	6	4	24
Score						

L2: Hauptadministrationsbüroangestelltenhierarchie

In the year 1910, the Austro-Hungarian administration in Prague is thriving, reaching the pinnacle of its efficiency. The grand building of the Imperial-Royal Administrative Office towers over the city. With modern technologies such as the telegraph, telephones, and typewriters, the work of clerks is becoming increasingly efficient. The time required to reject applications and stamp them "ABGELEHNT" has been reduced to just a few working days.

However, not everything is going according to plan. Due to a troubling systemic error, the percentage of positively reviewed applications has risen to an alarming 2.3%. The Ministry of the Interior in Vienna has deemed this a scandal and a threat to the bureaucratic order of the Monarchy. To tighten procedures and more effectively eliminate undesirable applications, a decision was made to implement a new administrative structure.

The new organizational model stipulates that each document must pass through a multi-stage verification system. In practice, this means that before a petitioner receives a missing passport page, their case must be passed through countless desks, flipped through, stamped, and ultimately shelved in the archives, where it will disappear forever.

To ensure the effectiveness of the new procedure, the Bureau of Administrative Modernization decided to implement a test system based on the latest advancements in computing machines. The entire simulation will be carried out using a multi-process computing architecture, and the flow of documents will be modeled using a system of signals exchanged between processes representing individual clerks.

The goal of the test is to create the perfect bureaucratic machine in which no application is ever approved, and every decision remains in indefinite suspension.

Remember to check for errors reported by system functions and to release allocated resources such as memory and file descriptors.

Stages:

1. [3 p.] Upon startup, the process verifies that it has been provided with two arguments. The first argument is the path to a directory containing files representing clerks. The second is the name of the highest-ranking clerk. These arguments are not yet used at this stage. The process should be started using
   `./sop-admin tree Franz_Joseph`.
   The process sets a signal mask blocking `SIGUSR1`, `SIGUSR2`, and `SIGINT`, prints `Waiting for SIGUSR1`, and waits for `SIGUSR1` to be delivered. Upon receiving it, the process terminates.

2. [7 p.] This entire stage should be contained within a single function. This function will be called recursively in later stages. It must take at least one argument: `char* name`, which is the name of a file in the directory specified as the first argument of the program. This function first prints
   `My name is <name> and my PID is <PID>`,
   opens the file `<argv[1]>/<name>`, and reads its contents using low-level API. The filename corresponds to the name of the clerk represented by this file. The file always contains two lines of text. Each line contains the name of the clerk's subordinate, or the character `-`, which represents the absence of a subordinate. For each subordinate defined in the file, the function prints
   `<filename> inspecting <subordinate's name>`.
   Then it prints
   `<filename> has inspected all subordinates`,
   and terminates.

3. [4 p.] After printing
   `<filename> inspecting <subordinate's name>`,
   in the previous stage, call `fork()`. The parent process continues working with the remaining subordinates, while the child process calls the function implemented in the previous stage, passing the

subordinate's name as an argument. The expected outcome of the program at this stage is the creation of a hierarchy of processes reflecting the organizational structure of the administration. The assignment sheet includes a graphical representation of the structure established in the `tree` directory.

After processing all subordinates, the process waits for their termination, prints
`<name> is leaving the office`,
and then terminates. Call the written function from `main`.

4. [6 p.] Implement the document processing logic. Instead of terminating after processing all subordinates, the process begins waiting for the `SIGUSR2` signal. Upon receiving it, it randomly chooses one of two actions. With a probability of $\frac{1}{3}$, it prints
`<name> received a document. Sending to the archive`,
and continues waiting for signals. With a probability of $\frac{2}{3}$, it prints
`<name> received a document. Passing on to the superintendent`,
sends `SIGUSR2` to its parent process, and continues waiting for signals. In the latter case, if the process is the highest-ranking clerk, it only prints `<name> received a document. Ignoring`.

5. [4 p.] Implement the office closing logic. When any process receives the `SIGINT` signal, it prints `<name> ending the day.`,
sends `SIGINT` to its children, waits for them to terminate, prints `<name> is leaving the office`, and then terminates.