Development Environment

OS: Ubuntu 22.04.3
GCC Version: 11.0.4

Implementation Details

The main function has 3 function calls. They are inputHandler(), scheduler(), and outputHandler().

inputHandler() is responsible to call necessary functions to read instructions.txt and definition.txt. Their content is read and instruction and process struct types are initialized respectively and kept in global arrays of struct. Lastly the instructions of the processes are read from their files and instructions array of the process struct is written.

scheduler() is the main body. It is responsible to iterate over the time. It calls pickNextProcess(int time) function to pick the process with the highest priority. If there is no process to run (pickNextProcess returns null) then it increase the time one unit. If there is one to run then it calls executeProcess(Process process, int time) function and it returns the finish time of execution. Then scheduler() takes the new time and repeats the same logic. It also responsible to handle context switches by adding 10 to the time when necessary. Also it calls the updateType(process) function after each execution to handle type conversions.

pickNextProcess(int time) iterates over the global array of processes and compares them one by one with compareProcesses(Process* a, Process *b) function. compareProcess has the logic to decide which one of them has priority over another and returns positive or negative number accordingly. Ultimately pickNextProcess(int time) returns which process must be executed at given time or null if none.

executeProcess(Process process, int time) function iterates over the instructions of the given process and checks whether to continue execution after each of them. There are three different condition to satisfy to continue execution: no preemption required, process has remaining instructions, and time quantum has not exceeded. It returns the finish time of execution if one of these conditions does not hold. This function also updates the corresponding field of the process struct with the time the process enters to the ready queue. Moreover, it updates the finish time if the process has no more instructions. (Finish time is initialized to be zero at the beginning so that it is also used to differentiate processes finished and not)

executeProcess(Process process, int time) function checks if preemption is required by calling ifPreemptionRequired(Process *executingProcess, int currentTime) function. It iterates over the global array of processes and compares them with executing process by calling preemptiveCompareProcesses(const Process *newProcess, const Process *executingProcess). This function returns positive number if the newProcess can preempt executingProcess. Then ifPrememptionRequired function returns 1 if any preemption is required and 0 otherwise.

Finally outputHandler() calculates and prints average waiting and turnaround times.