Process Scheduler

This project involves creating a Process Scheduler for a virtual Operating System, using Linked Lists. In an Operating System (OS), each program that is executed is called a Process while it is running. Each process has a unique value called a Process ID (PID), which is how the OS tracks and manages each process. The Program Implements a Process Scheduler for the PID's and performs a context switch when required.

The functions implemented for the scheduler are simulated OS and CPU. For the project, each process will be represented as a struct and the schedule list will be a singly linked list implemented in C. The functions will be called by the OS to create processes and fill them in with provided initial values, insert processes into the list in order, select a process to run and remove it from the list, free processes, and provide a count of all of the processes in the scheduler list.

The CPU scheduling algorithm implements a Shortest Remaining Time First. This means every time we need to select a new process, we choose the one with the shortest time remaining value. Functions implemented and their description:

1. **Counts the Items in the List**
   int schedule_count(Process *list)
   - This function counts all of the processes in the linked list called list. The head node of this list is passed in as the one parameter for this function.
   - Returns the number of processes in the list.

2. **Inserts an Existing Node into the List**
   void schedule_insert(Process **list, Process *node)
   - This function inserts the Process node into the linked list. The head node of the list is passed in as a Process ** type double-pointer.
   - Inserts the given node in order of ascending pid value.

3. **Free an Existing Node not in the List**
   void schedule_terminate(Process *node)
   - This function frees all memory associated with the given node.

4. **Creates a New Node from the Given Parameters**
   Process *schedule_generate(const char *name, int pid, int time_remaining, int time_last_run)
   - This function creates a new node of type Process * using dynamic memory allocation. Initializes all members of the new Process struct to the values provided.
   - Since name is a string, uses a string function to copy name into the new struct. Uses strncpy to perform this copy and returns NULL on any memory errors (malloc) return.

5. **Selects a Node and Removes it from the List**

Process *schedule_select(Process **list)

- This function will perform three tasks.
    i. It will select the next process to run according to a set of rules.
    ii. It will remove that process from the list and adjust the list if needed.
    iii. It will return the selected process.
- Selects the process from the list using the following two rules:
    i. Selects the process with the lowest time_remaining value. If there is a tie, chooses the process with the lowest pid value.
    ii. If there is any process that has not run in >= TIME_STARVATION time, selects that process instead of the one with the lowest time_remaining value.
- If there are multiple processes that are in Starvation, selects the one with the lowest pid value.
- Removes the selected process from the list. Updates the list if the one removed was the beginning of the list. Returns the selected process or returns NULL if list is empty.

OUTPUT

```
vbandi@zeus-2:~/beta/handout$ ./scheduler
.=============
| Starting Time:  1
+------------
| Process Starting
|       PID: 886 Time Remaining:  5 (Last Run Time:  1) Name: cp
+-----------
| In Scheduler:  1 Processes
|       PID: 886 Time Remaining:  5 (Last Run Time:  1) Name: cp
+------------
| Selecting to Run on the CPU
|       PID: 886 Time Remaining:  5 (Last Run Time:  1) Name: cp
\=============

.=============
| Starting Time:  2
+------------
| Unloading Process
|       PID: 886 Time Remaining:  4 (Last Run Time:  1) Name: cp
| Process Starting
|       PID: 335 Time Remaining:  2 (Last Run Time:  2) Name: nano
+------------
| In Scheduler:  2 Processes
|       PID: 335 Time Remaining:  2 (Last Run Time:  2) Name: nano
|       PID: 886 Time Remaining:  4 (Last Run Time:  1) Name: cp
+------------
| Selecting to Run on the CPU
|       PID: 335 Time Remaining:  2 (Last Run Time:  2) Name: nano
\=============
```

```
.=============
| Starting Time:  8
+------------
| Terminating Process
|       PID: 886 Time Remaining:  0 (Last Run Time:  7) Name: cp
| Process Starting
|       PID: 135 Time Remaining:  2 (Last Run Time:  8) Name: mkdi
+------------
| In Scheduler:  6 Processes
|       PID:  59 Time Remaining:  6 (Last Run Time:  4) Name: touc
|       PID: 135 Time Remaining:  2 (Last Run Time:  8) Name: mkdi
|       PID: 368 Time Remaining:  9 (Last Run Time:  6) Name: uniq
|       PID: 421 Time Remaining:  7 (Last Run Time:  3) Name: emacs
|       PID: 426 Time Remaining:  6 (Last Run Time:  5) Name: grep
|       PID: 530 Time Remaining:  3 (Last Run Time:  7) Name: emacs
+------------
| Selecting to Run on the CPU
|       PID: 421 Time Remaining:  7 (Last Run Time:  3) Name: emacs
\=============

.=============
| Starting Time:  9
+------------
| Unloading Process
|       PID: 421 Time Remaining:  6 (Last Run Time:  8) Name: emacs
| Process Starting
|       PID:  58 Time Remaining:  7 (Last Run Time:  9) Name: whoam
+------------
| In Scheduler:  7 Processes
|       PID:  58 Time Remaining:  7 (Last Run Time:  9) Name: whoam
|       PID:  59 Time Remaining:  6 (Last Run Time:  4) Name: touc
|       PID: 135 Time Remaining:  2 (Last Run Time:  8) Name: mkdi
|       PID: 368 Time Remaining:  9 (Last Run Time:  6) Name: uniq
|       PID: 421 Time Remaining:  6 (Last Run Time:  8) Name: emacs
|       PID: 426 Time Remaining:  6 (Last Run Time:  5) Name: grep
|       PID: 530 Time Remaining:  3 (Last Run Time:  7) Name: emacs
+------------
| Selecting to Run on the CPU
|       PID:  59 Time Remaining:  6 (Last Run Time:  4) Name: touc
\=============
```