Test case documentation - Practical exercise 1 - Gruppe 88

Test case 1 - Schedule and list

This tests simply creating an alarm process, checking if it goes off at the specified time and checking if the process is terminated.

1. Create an alarm in 2 minutes
2. Check that the number of seconds until the alarm goes off is correct (about 120 seconds)
3. List alarms using the command “l” to see if it appears
4. Wait 2 minutes to see if the alarm goes off
5. List alarms to see if it is removed from the list
6. Call “ps -e” in another terminal to ensure that the zombie alarm is removed (only the parent process running)
7. Exit the process using “x”

Test case 2 - Schedule and cancel

This test is designed to check how the program handles multiple alarms and if the alarms are saved as the user intended. It also checks the behavior of the program when deleting an alarm; if the alarm is removed from the overview of alarms and the alarm does not go off.

1. Create an alarm in 2 minutes
2. Create an alarm in 3 minutes
3. Create an alarm in 4 minutes
4. List alarms to see that all 3 alarms were scheduled.
    a. Ensure that the time of each alarm is the same as inputted
5. Input “c” to cancel an alarm. Choose ID=2 to delete the alarm you scheduled in 3 minutes
6. List alarms to see that there now is 2 alarms scheduled
    a. Ensure that the two alarms listed is the one you scheduled in 2 minutes and 4 minutes by looking at the time they are said to go off
    b. The alarm names should now be “Alarm 1” and “Alarm 2”
7. Call “ps -e” to ensure that there are the correct number of alarm processes
    a. The output should be three processes with the same name as the executable (for instance a.out)
8. Wait 3 minutes to ensure that the alarm you scheduled in step 2 does not go off
9. Wait till the alarm you scheduled in step 3 goes off
10. List alarms to see if every alarm is removed from the list
11. Call “ps -e” in another terminal to ensure that remaining zombie alarms are removed

    a. The output should now be only one process named the same as the executable (for instance a.out)
12. Exit the process using “x”

Test case 3 - Maximum number of alarms

This tests that the program behaves as expected when the maximum number of alarms is reached. When 10 alarms are scheduled and the user tries to schedule an additional alarm, the program is expected to inform the user that the maximum number of alarms is reached. If an alarm is removed from the array, by either going off or being canceled by the user, the user may add one more alarm.

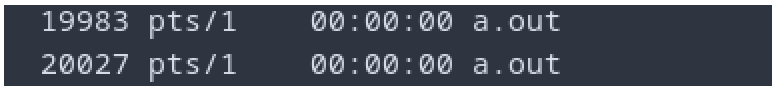
1. Schedule 10 (maximum number) alarms
2. Try to schedule an additional alarm to ensure that the correct feedback is given
    a. Expected behaviour: When inputting s and you have 10 alarms queued you should be notified that the maximum number of alarms is reached and that you need to cancel one or wait for it to ring.
3. Cancel one alarm using “c”
4. List alarms to see that there now is room for another alarm
5. Schedule one more alarm
6. List alarms to see that the new alarm was added and the array is full
7. Wait for an alarm to go off
8. List alarms to see that there now is room for another alarm
    a. Should be 9 alarms
9. Schedule one more alarm
10. Exit process using “x”

Test case 4 - Process management

This test is designed to test how the program handles processes. It tests that when starting the program one process is started, and when scheduling an alarm the program forks a child process. After an alarm is rung, the child process becomes a zombie process. The test checks that these zombie processes are cleared when the user inputs a command in the starting menu. It also checks that the parent process of all the alarms (the program itself) is terminated when the user wants to exit the program.

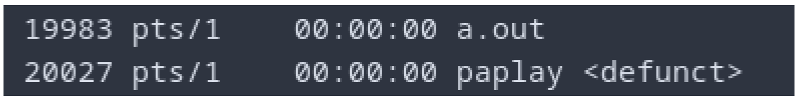
1. Start the executable
2. Call “ps -e” and ensure that we have one process now named a.out (or your executable name)
3. Schedule an alarm in 2 minutes
4. Call “ps- e” to see that the alarm was scheduled. You should now have two processes with the same name as the binary executable (see Picture 1).

5. Wait till the the alarm you scheduled in step 3 is ringing
6. Call “ps -e” to see that one of the processes is marked as “<defunct>” (see Picture 2). This indicates that we have a zombie process
7. Try to input one either ‘l’, ‘s’ or ‘c’ along with ENTER to the alarm clock. If the program works as expected this should clear all zombie child processes
8. Call “ps -e” to ensure that the process named “defunct” now is not listed.
9. Exit the process using “x”
10. Call “ps -e” to ensure that the parent process is now terminated



```
 19983 pts/1    00:00:00 a.out
 20027 pts/1    00:00:00 a.out
```

Picture 1: *The expected output after scheduling an alarm if your executable is named a.out. You can safely ignore the processIDs and process 18844.*



```
19983 pts/1    00:00:00 a.out
20027 pts/1    00:00:00 paplay <defunct>
```

Picture 2: *The expected output after the alarm has rung. Notice that the process with process ID 20027 now has the cmd value paplay. This is because the alarm has executed the paplay process to play sound.*