Лабораторная работа II

Мониторинг процессов и ресурсов в Linux

В рамках данной лабораторной работы можно предоставить реализацию всех скриптов как на Bash, так и на языке Си. *При выборе второго варианта вы можете претендовать на 150% от максимальной стоимости задания.*

Напишите скрипты, решающие следующие задачи:

1. Посчитать количество процессов, запущенных пользователем `$USER`, и вывести в файл получившееся число, а затем пары `PID:команда` для таких процессов
2. Вывести в файл список `PID` всех процессов, которые были запущены командами, расположенными в `/sbin/`
3. Вывести на экран `PID` процесса, запущенного последним (с последним временем запуска)
4. Для всех зарегистрированных в данный момент в системе процессов определить среднее время непрерывного исполнения процессора (`CPU_burst`) и вывести в один файл строки `ProcessID=PID : Parent_ProcessID=PID : Average_Running_Time=ART`

   Значения `PPid` можно взять из файлов `status`, которые находятся в директориях с названиями, соотвествующими `PID` процессов в `/proc`. Значения `ART` получить, разделив значение `sum_exec_runtime` на `nr_switches`, взятые из файлов `sched` в этих же директориях. Отсортировать эти строки по идентификаторам родительских процессов.
5. В полученном на предыдущем шаге файле после каждой группы записей с одинаковым идентификатором родительского процесса вставить строку вида `Average_Running_Children_of_ParentID=N is M`

   где N = PPID, а M – среднее, посчитанное из ART для всех процессов этого родителя.
6. Используя псевдофайловую систему `/proc` найти процесс, которому выделено больше всего оперативной памяти. Сравнить результат с выводом команды `top`.
7. Написать скрипт, определяющий три процесса, которые за 1 минуту, прошедшую с момента запуска скрипта, считали максимальное количество байт из устройства хранения

данных. Скрипт должен выводить `PID`, строки запуска и объем считанных данных, разделенные двоеточием.

Правила оформления и написания программ на C

Программы должны удовлетворять следующим требованиям:

- В случае, если это требуется по ТЗ, корректно обрабатывать ошибки при взаимодействии с внешним миром: ошибки ввода-вывода, некорректный пользовательский ввод и прочее. Если это произошло – необходимо вывести сообщение об ошибке (на английском языке) и завершить исполнение с ненулевым кодом возврата.
- Программа никогда не должна падать. Падение – признак ошибок в реализации.
- Программа не должна содержать лишних сущностей: закомментированных больших участков кода, неиспользуемых переменных и функций и тому подобное. Это засоряет код и увеличивает время проверки.

При успешном выполнении программа возвращает код `0`. Если же что-то пошло не так, то она сообщает о проблеме через ненулевой код возврата и сообщение об ошибке в поток вывода ошибок `stderr`.

За что можно потерять баллы к критерии качества кода

Кроме правильности результата будет учитываться скорость работы программы. То есть, если проверяющий не дождался за разумное время завершения работы программы, то тест будет считаться не пройденным.

В программе можно использовать стандартные библиотеки и заголовочные файлы, а также те, которые относятся к POSIX библиотеке. Например, `<bits/stdc++.h>` таковым не является и его использование влечёт за собой потерю баллов.

Если программа использует функции, которые явно не объявлены в файле с исходным кодом (например, тип `size_t` без подключения `<stdlib.h>` и пр.), то за это также будут снижаться баллы (даже если у вас всё работает).