САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: OS lite

Отчет

по домашней работе №5

**«Управление памятью в ОС Linux»**

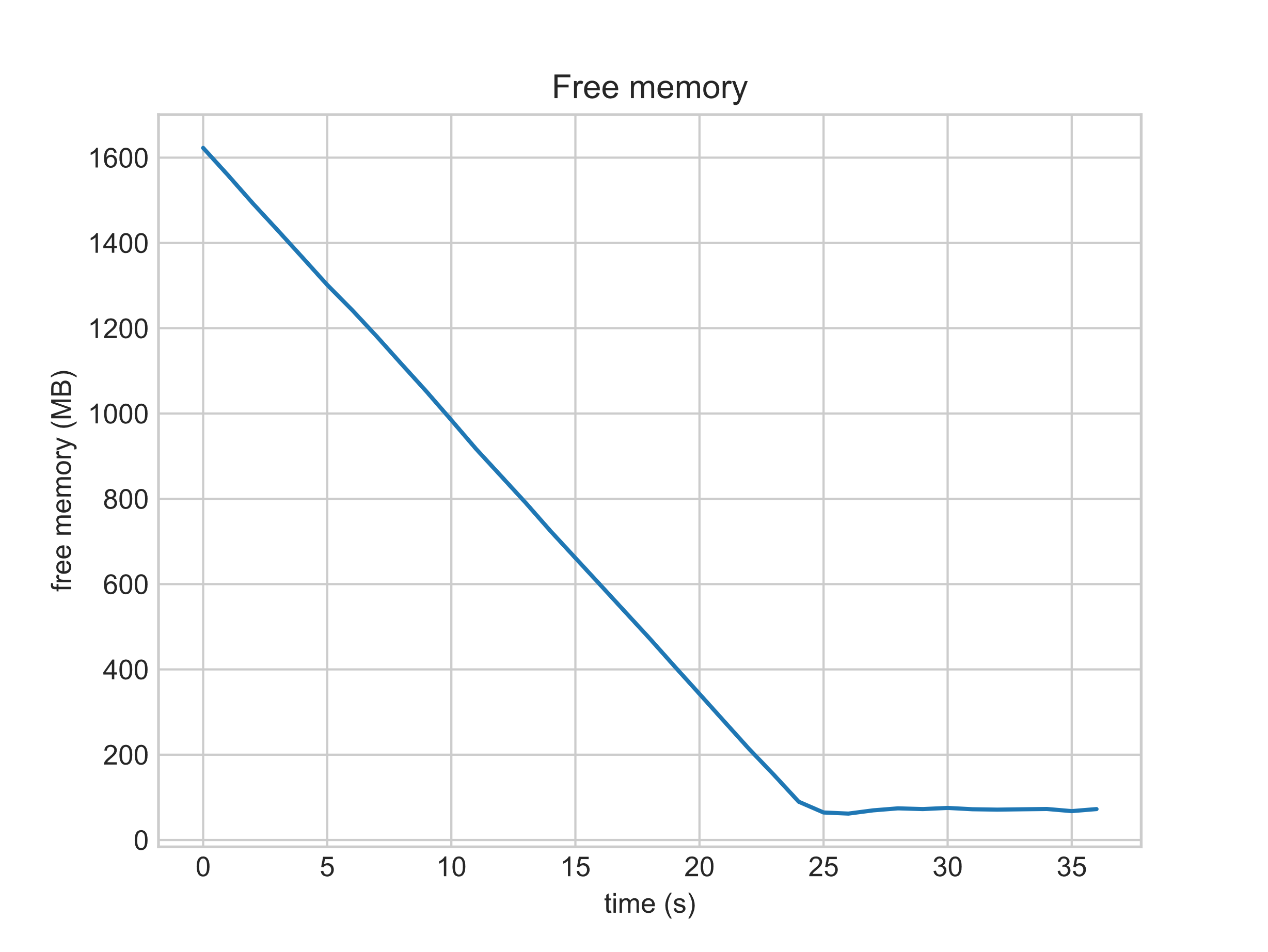
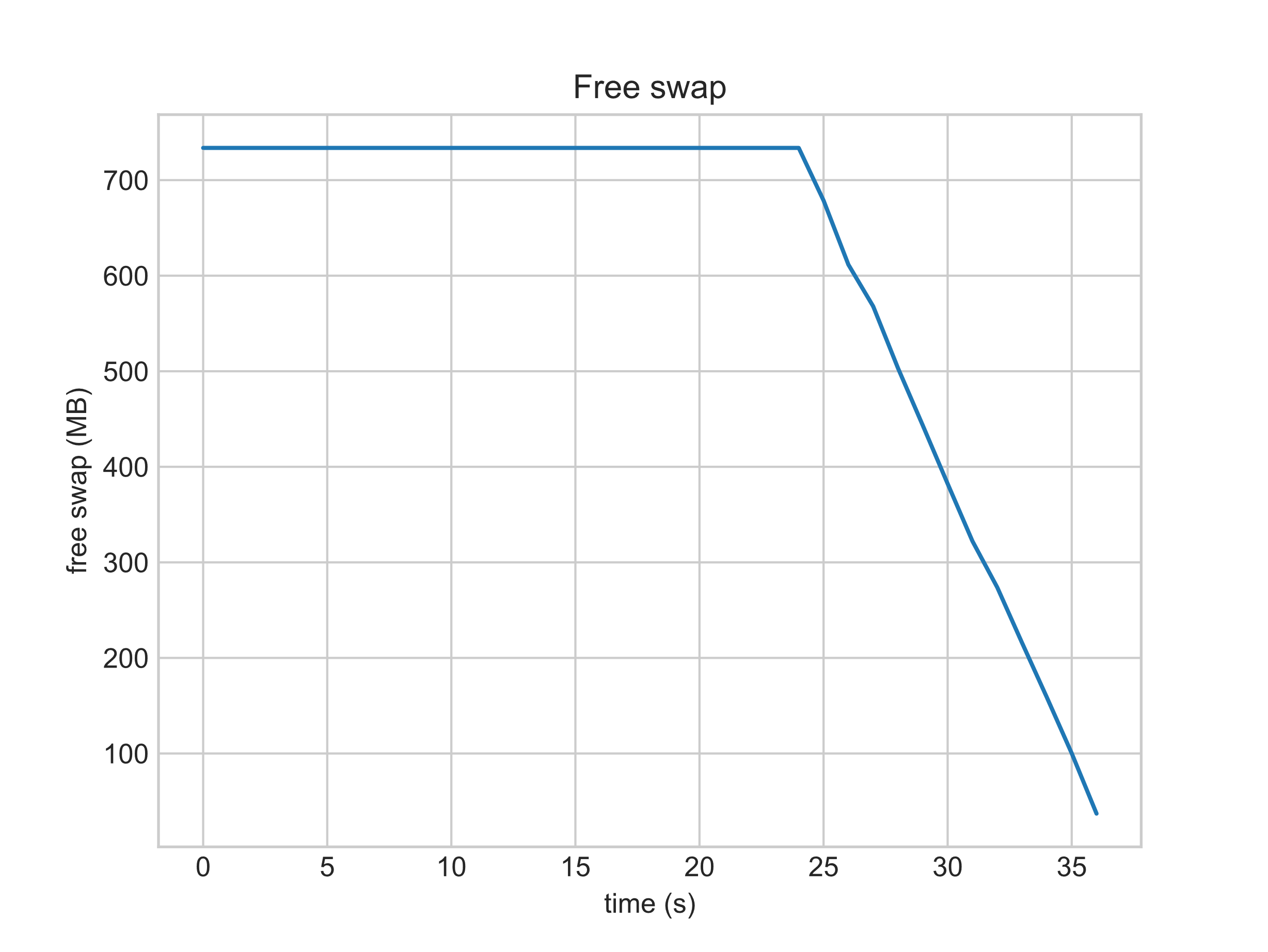
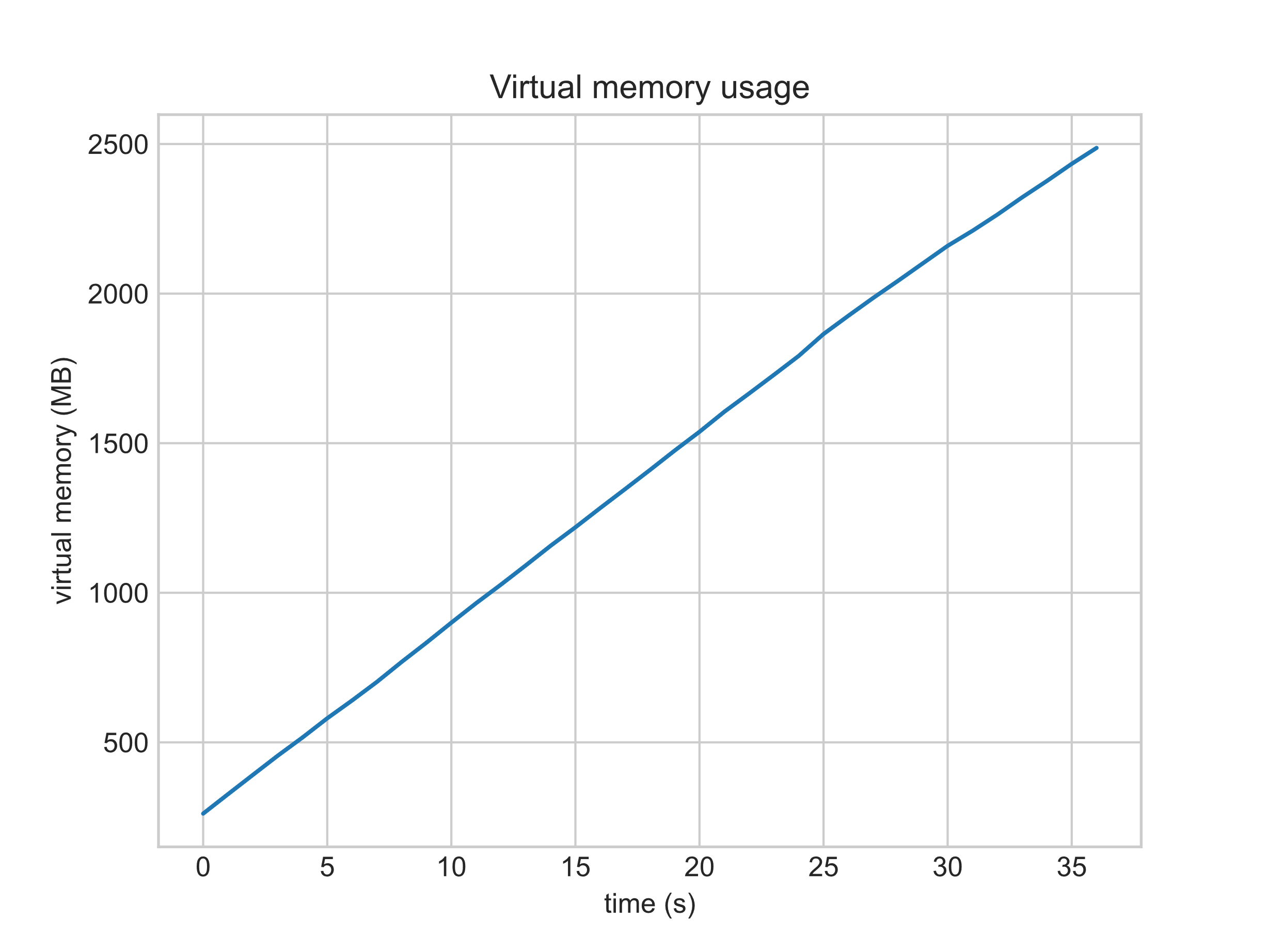
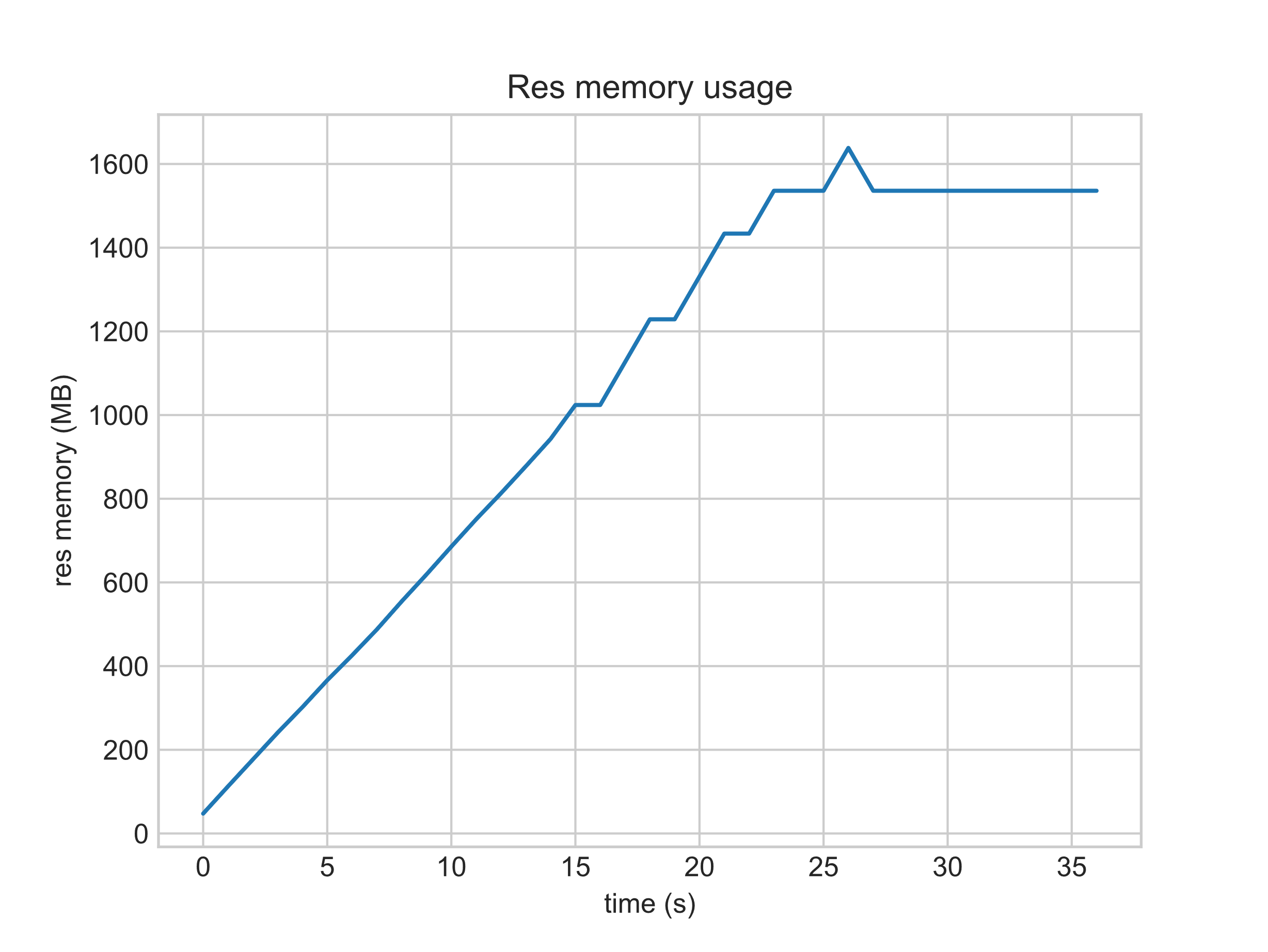
Выполнил(а): Ступников Александр Сергеевич

студ. гр. M3235

Санкт-Петербург

2020

**Эксперимент №1  
Первый этап**

********

По приведённым графикам видно, что по мере выполнения скрипта общий объём памяти (график «Virtual memory usage»), потребляемый mem.bash (далее скрипт), линейно растёт. Притом сначала (примерно до 24 секунды с начала измерений) скрипту выделяется физическая память, страничного обмена не происходит. Далее физическая память заканчивается и начинается страничный обмен. По графику «Free swap» видно, что объём свободной памяти в swap не меняется до 24 секунды, далее же он линейно уменьшается по мере того, как происходит страничный обмен. Заметим, что после начала страничного обмена объём свободной физической памяти не меняется (график «Free memory»), также не меняется объём физической памяти, которую использует скрипт (график «Res memory usage»).

Как только объём свободной физической памяти, а также объём свободной памяти в swap принимает значение близкое к 0 (примерно 36 секунда с момента начала измерений), скрипт падает с ошибкой о нехватке памяти:

**[21456.435111] Out of memory: Killed process 80750 (mem.bash) total-vm:2597724kB, anon-rss:1626660kB, file-rss:0kB, shmem-rss:0kB, UID:1000 pgtables:4724kB oom\_score\_adj:0**

Видно, что общий объём виртуальной памяти, выделенной скрипту, когда тот аварийно завершил выполнение, равен **2597724kB**, что примерно соответствует сумме объёмов свободной физической памяти и свободной памяти в swap, указанных в meminfo. Последняя строчка файла report.log содержит число 3000000, то есть скрипт смог создать в памяти массив размером 3000000 \* 10 = 30000000.

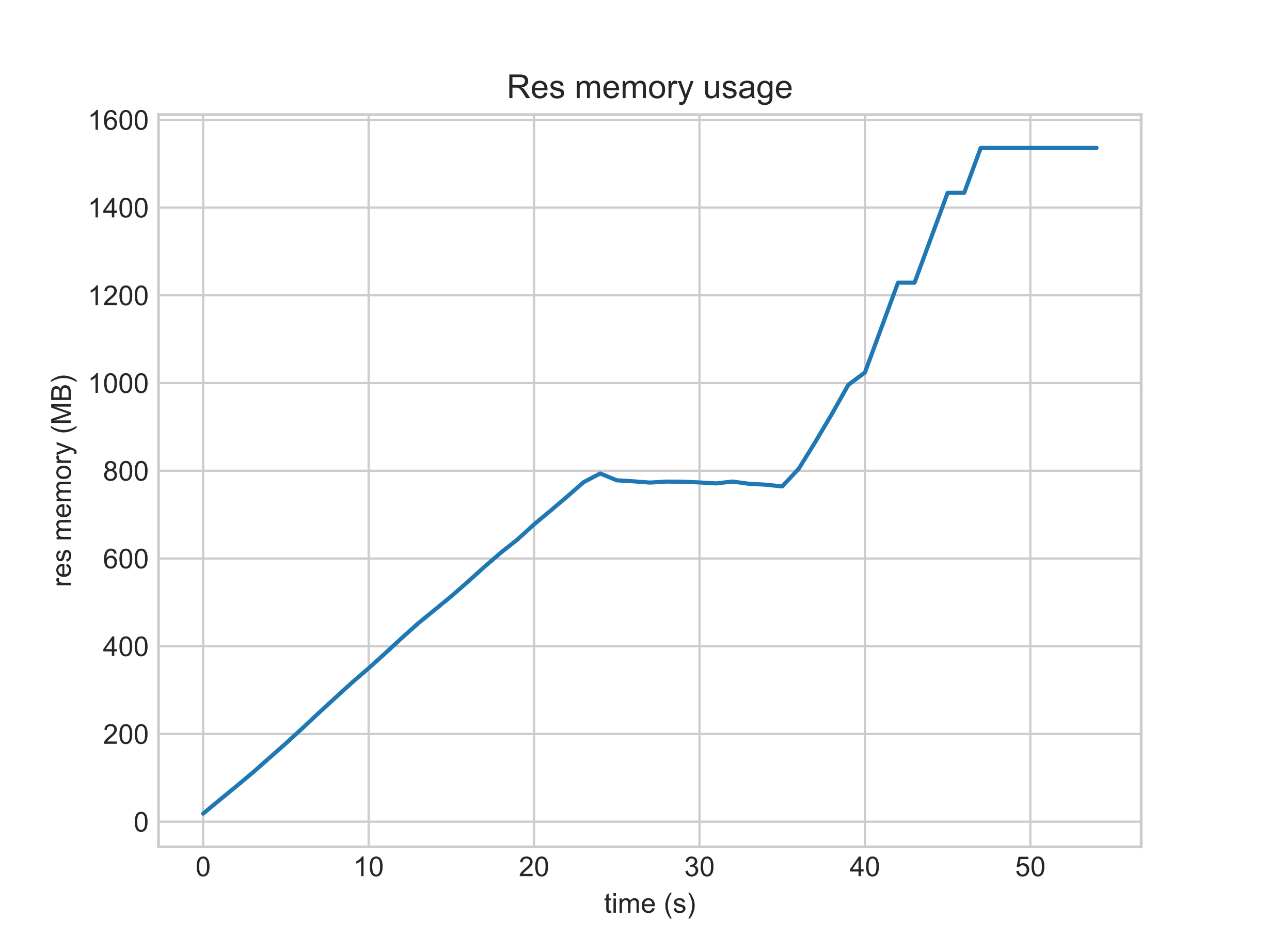
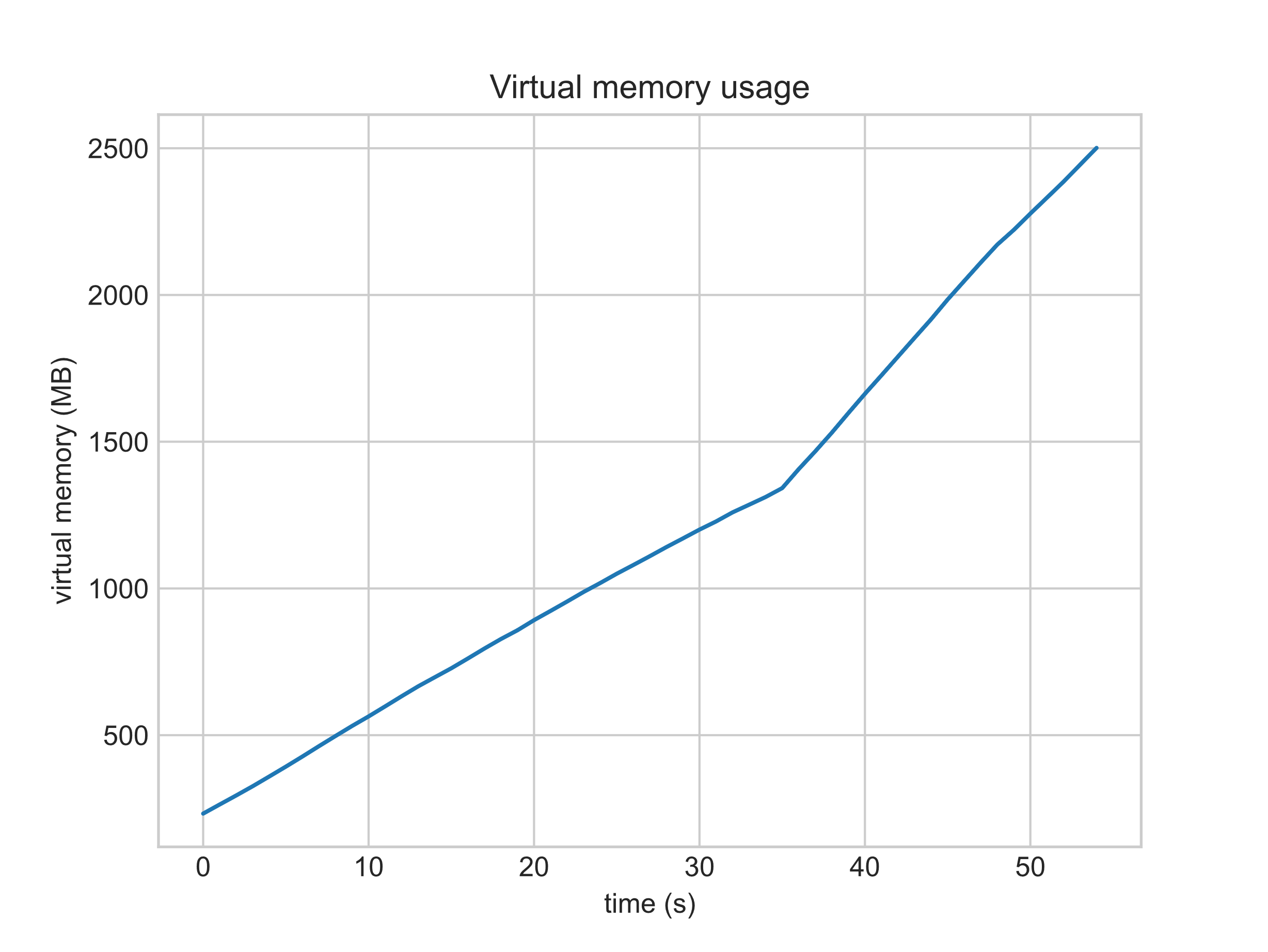
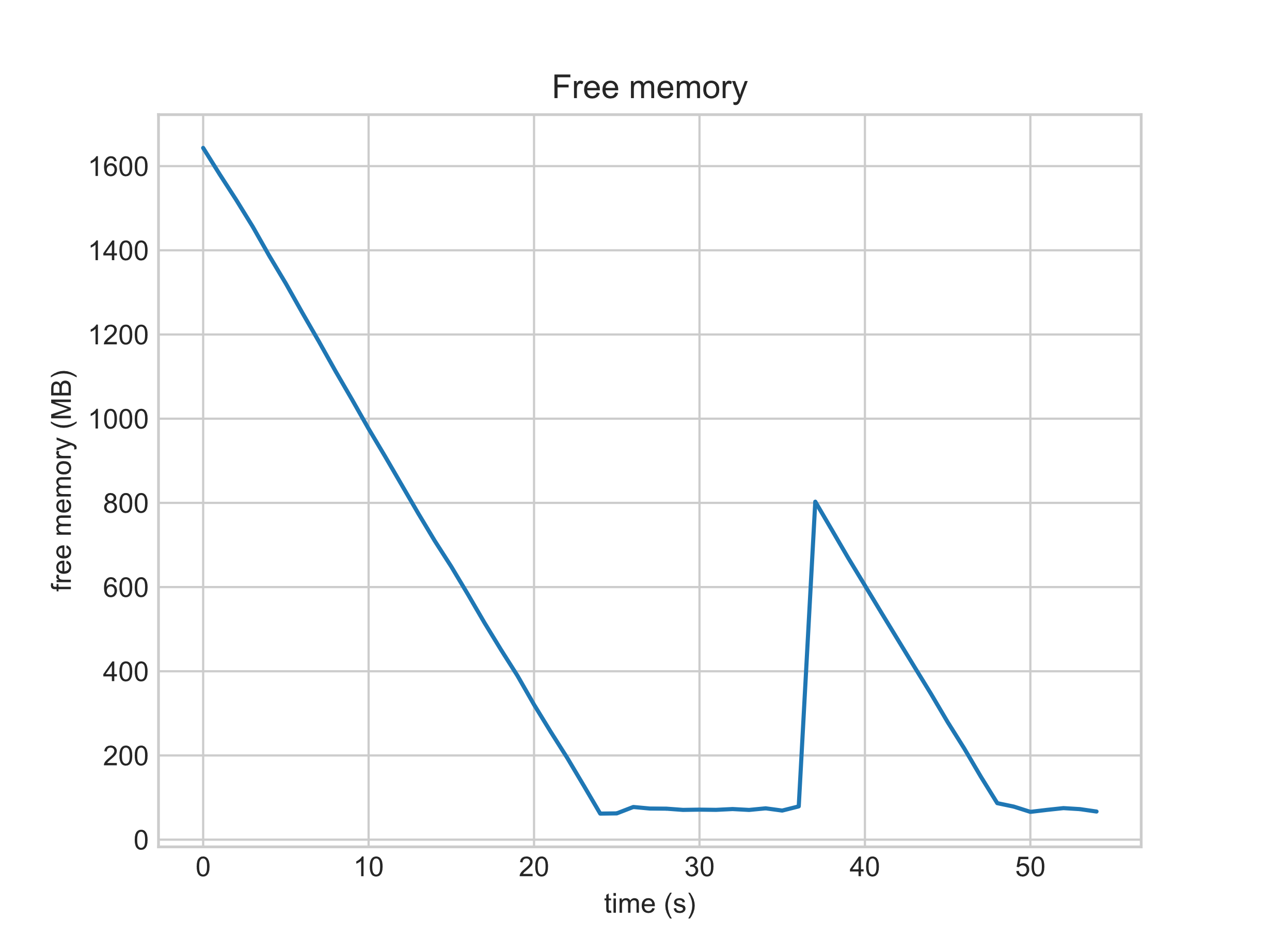
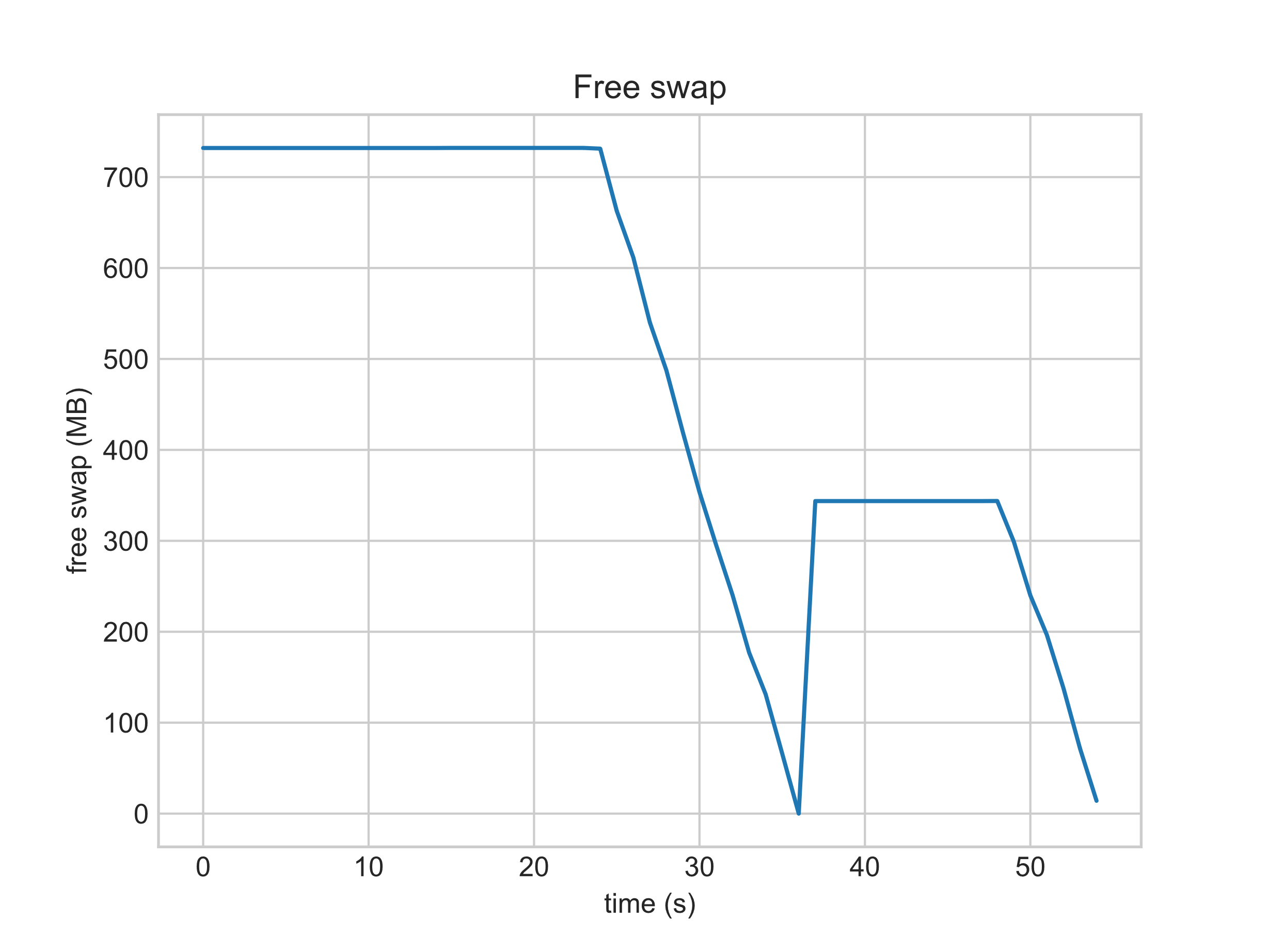
Следует заметить, что на протяжении всего времени выполнения скрипта список процессов, потребление памяти которых максимально, почти не меняется. Этот список выглядит примерно следующим образом:

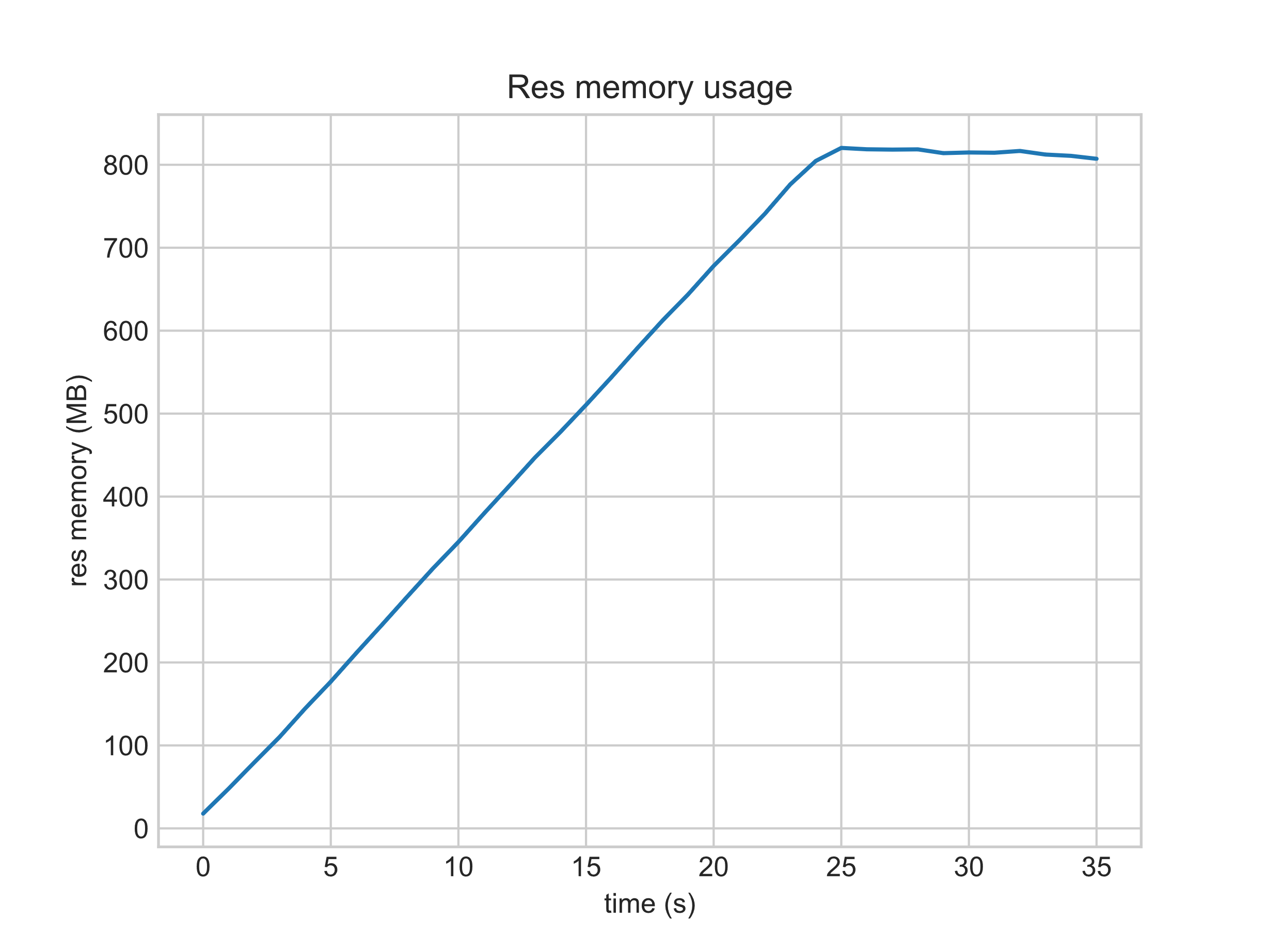
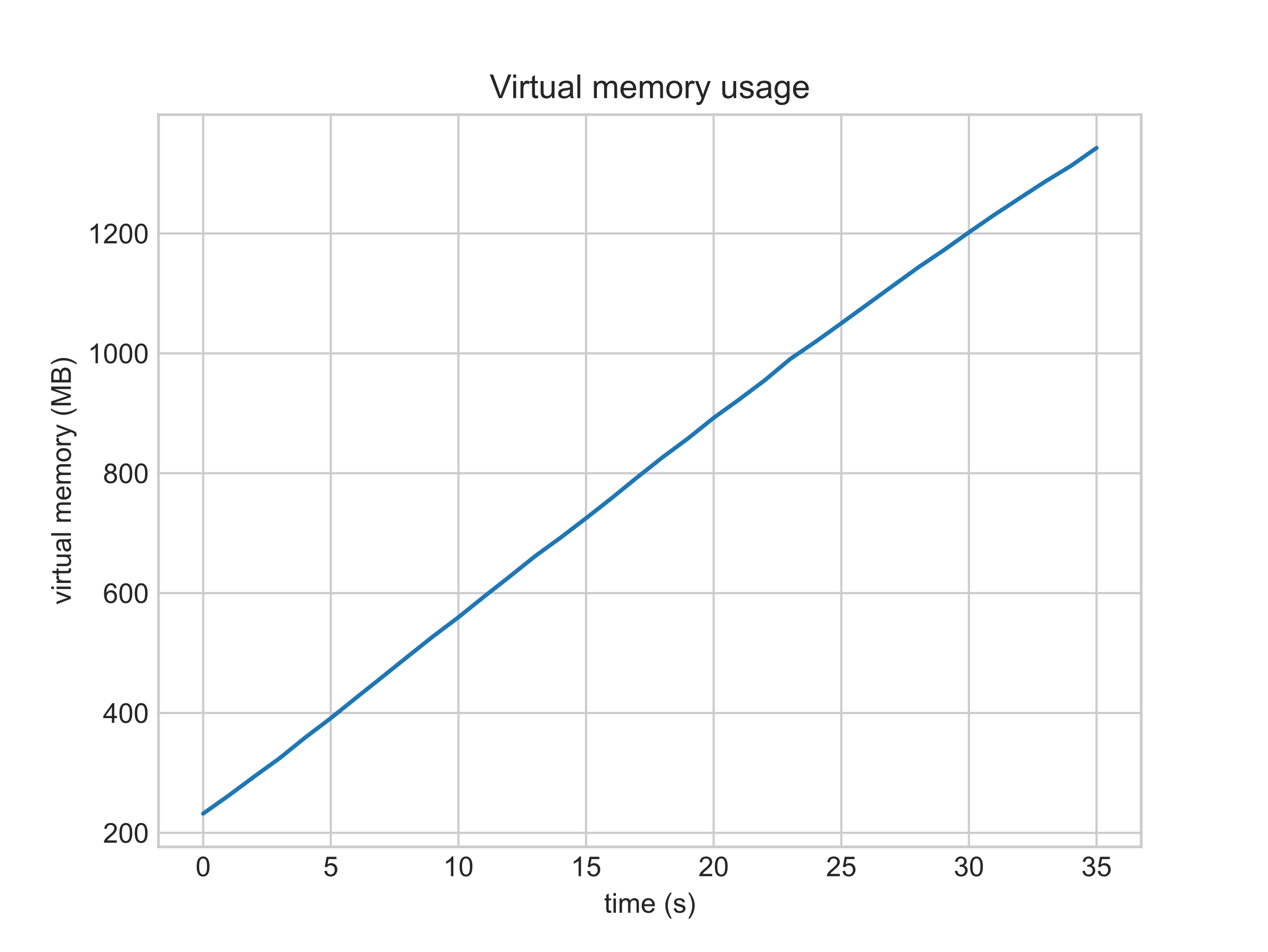
1. mem.bash
2. systemd-journal
3. systemd
4. top
5. awk

Притом сам скрипт неизменно занимает в этом списке первое место. Отмечу, что при аварийном завершении скрипта в top на короткий промежуток времени появляется процесс oomreaper. Этот процесс, запускающийся в отдельном потоке, освобождает память, которую использовал скрипт, в то время как сам скрипт, получив сигнал sigterm, завершает свою работу. В журнале появляется следующее сообщение:

**[21456.631661] oom\_reaper: reaped process 80750 (mem.bash), now anon-rss:0kB, file-rss:0kB, shmem-rss:0kB**

**Второй этап**

**mem1.bash**

**mem2.bash**

Изменение параметров, приведённых на графиках, на втором этапе эксперимента не слишком отличается от изменения таковых на первом этапе эксперимента. Отметим лишь те моменты, которые отличают второй этап эксперимента от первого.

Теперь одновременно запускаются два скрипа mem1.bash (далее скрипт 1) и mem2.bash (далее скрипт 2). Видно, что скорость уменьшения свободной физической памяти (график «Free memory») на втором этапе эксперимента не отличается от таковой на первом этапе эксперимента. Однако теперь скрипт 1 и скрипт 2 должны делить между собой возможности операционной системы по выделению памяти. Поэтому видно, что во время одновременного выполнения обоих скриптов увеличение объёма выделенной памяти (график «Virtual memory usage») для каждого конкретного скрипта происходит на втором этапе эксперимента примерно в два раза медленнее чем на первом. Также видно, что в top скрипты 1 и 2 постоянно меняются местами, занимая лидирующие позиции по объёму используемой памяти.

Интересен момент времени 35 с. В этот момент времени скрипт 2 аварийно завершает свою работу с сообщением:

**[21601.398118] Out of memory: Killed process 81346 (mem2.bash) total-vm:1418964kB, anon-rss:845176kB, file-rss:0kB, shmem-rss:0kB, UID:1000 pgtables:2416kB oom\_score\_adj:0**

Видно, что объём виртуальной памяти, выделенный скрипту 2 на момент его завершения примерно равен половине свободной памяти вообще (по данным из meminfo при спокойном состоянии системы). Скрипт 2 смог создать в памяти массив размером примерно 15000000, это половина длины массива, созданного скриптом на первом этапе эксперимента.

Скрипт 1 продолжает свою работу. Видно, что в рассмотренный момент времени после завершения работы скрипта 2, объём свободной физической памяти и объём свободной памяти в swap резко возрастает (графики «Free memory» и «Free swap»). Это обусловлено тем, что по завершении работы скрипта 2 память, выделенная ему, освобождается и может быть использована скриптом 1. По графику «Virtual memory usage» для скрипта 1 видно, что скорость выделения памяти этому скрипту в момент времени 35 секунд, резко возрастает примерно в 2 раза, это происходит, потому что теперь скрипт 1 больше не борется со скриптом 2 за выделение памяти. По графику «Res memory» для скрипта 1, а также по графику «Free swap» можно сделать вывод о том, что в момент времени 35 секунд страничный обмен прекратился, так как стало доступно место в физической памяти. В момент времени 46 секунд страничный обмен вновь возобновляется, так как место в физической памяти заканчивается.

В итоге примерно на 55 секунде после начала выполнения скрипт 1 аварийно завершает работу с сообщением об ошибке:

**[21642.077554] Out of memory: Killed process 81349 (mem1.bash) total-vm:2607360kB, anon-rss:1637548kB, file-rss:0kB, shmem-rss:0kB, UID:1000 pgtables:4752kB oom\_score\_adj:0**

Видно, что общий объём виртуальной памяти, выделенной скрипту 1, когда тот аварийно завершил выполнение, равен **2607360kB**, что примерно равно объёму виртуальной памяти, которая была выделена скрипту на первом этапе эксперимента на момент его аварийного завершения. Последняя строчка файла report1.log содержит число 3000000, то есть скрипт смог создать в памяти массив размером 3000000 \* 10 = 30000000. Это в точности равно размеру массива, который создал скрипт на первом этапе эксперимента, что неудивительно, учитывая, что по завершении работы скрипта 2 скрипт 1 ведёт себя аналогично скрипту из первого этапа эксперимента.

Следует заметить, что на протяжении всего одновременного выполнения скриптов список процессов, потребление памяти которых максимально, совершенно не меняется. Этот список выглядел примерно следующим образом:

1. mem1.bash
2. mem2.bash
3. systemd-journal
4. systemd
5. top

**Эксперимент №2**

При K=30 и n=3000000 суммарный максимальный объём памяти, необходимый процессам, если бы они все были запущены одновременно, равен общему объёму памяти ОС, увеличенному в 3 раза. Учитывая, что процессы запускаются с задержкой в 1 секунду, пиковая потребность в памяти у совокупности процессов несколько ниже, но всё равно больше, чем доступный объём памяти ОС. Поэтому некоторые процессы завершаются аварийно.

Если уменьшить n до 1750000, ни один из запущенных процессов не завершит свою работу аварийно. Это свидетельствует о том, что при таком n процессы завершают свою работу (не аварийно), тем самым освобождая выделенную им память, быстрее, чем другие процессы запрашивают память.