Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет Информационных технологий и управления
Кафедра Интеллектуальных информационных технологий

ОТЧЁТ

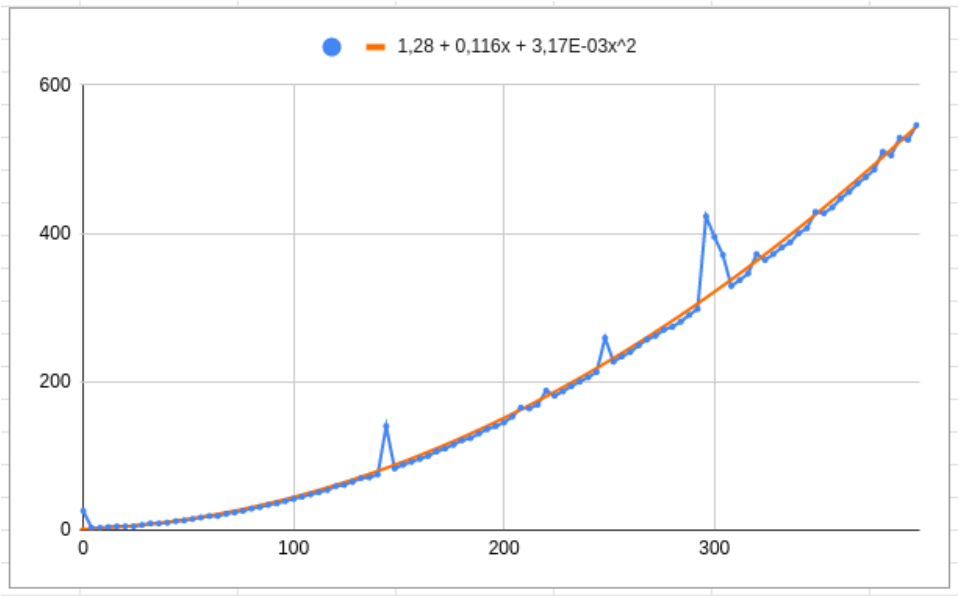
по лабораторной работе №4
по дисциплине “Операционные системы”

Выполнили:
Р. В. Липский, И. А. Жолнерчик, гр. 121701
Проверил:
В. А. Цирук

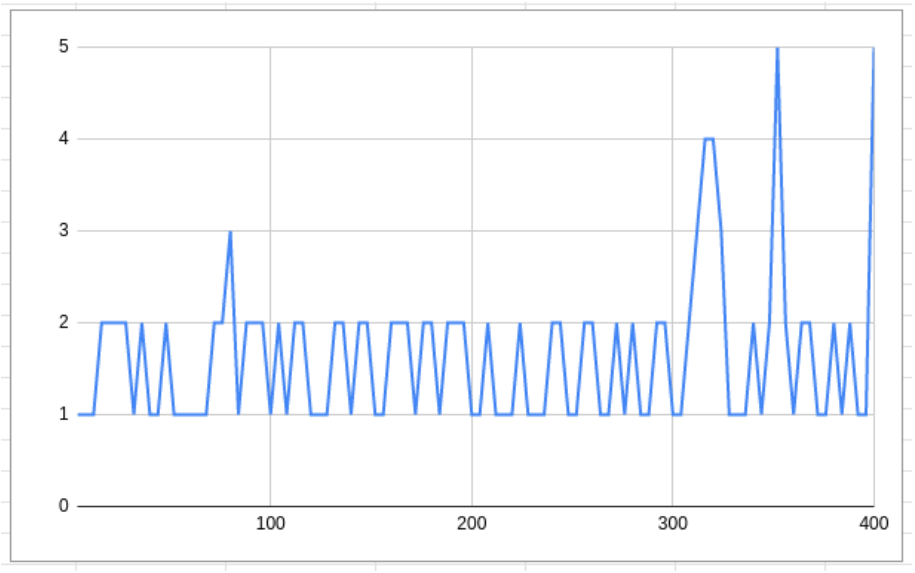
Цель: Изучить алгоритмы распределения памяти и особенности их реализации.

Задание: Реализовать менеджер памяти со страничным разбиением.

Нагрузочные тесты:
Время выделения памяти от количества уже выделенной памяти:



По оси Y измеряется время, затраченное на выделение памяти в микросекундах, по оси X количество уже выделенной памяти в байтах.
Синие точки и полоса - реальные данные, оранжевая - линия тренда.
График экспоненциальный из-за не очень эффективного алгоритма поиска свободного места - это три вложенных цикла. Из возможных более эффективных решений: построить дерево указателей на участки свободной памяти и искать по нему.
Время освобождения памяти от количества выделенной памяти:



Удаление записей о выделенной памяти имеют сложность O(1), поскольку используется структура данных “хэш-таблица”, удаление записей из которой имеет такую временную сложность.

Время выделения памяти от объема выделяемой памяти:

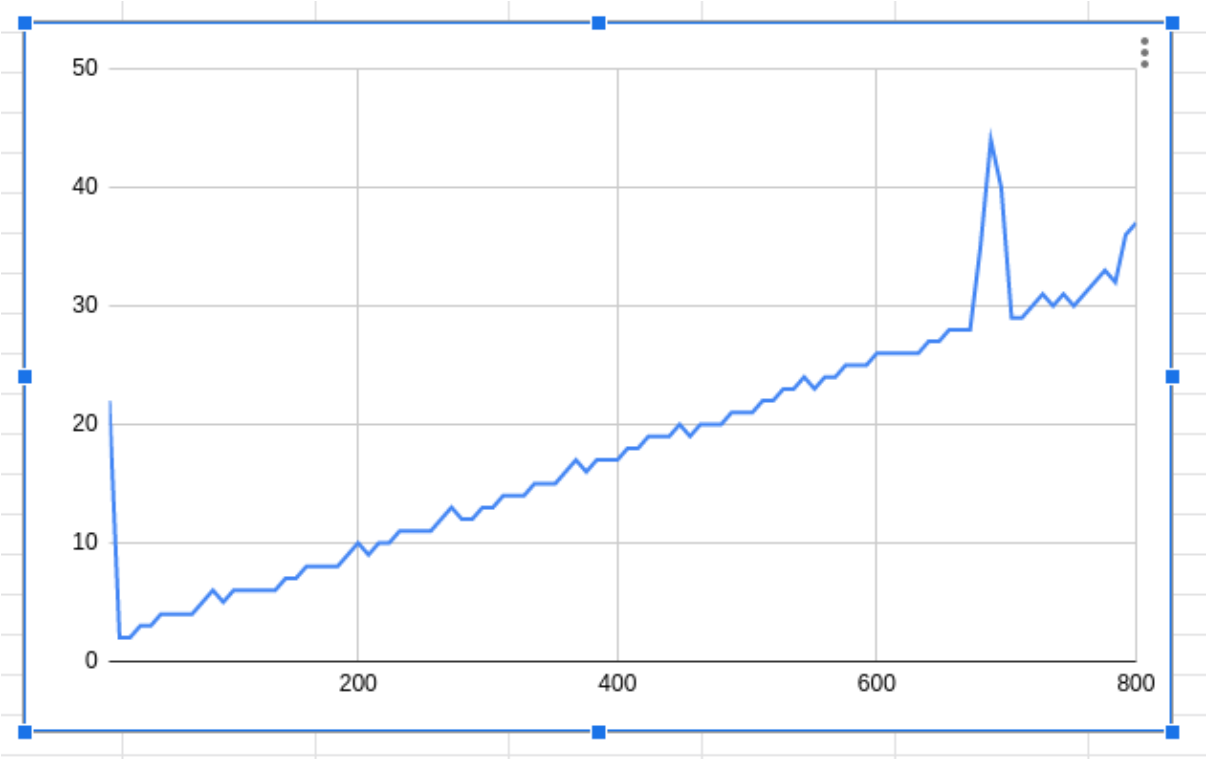


График растёт линейно, поскольку при пустой странице временная сложность выделения памяти O(n) - чем больше количество выделяемых байт, тем больше байт в памяти нужно проверить на то, свободны ли они.

Преимущества станичного выделения памяти:

- Отсутствие внешней фрагментации
- Легкая отгрузка давно не используемых страниц в область подкачки на диске
- Поддержка отображённых в память файлов
- Поддержка разделяемой между процессами памяти, в том числе с копированием-по-записи для экономии физических страниц

Недостатки страничного выделения памяти:

- Внутренняя фрагментация
- Процессам может быть нужны размеры, некратные размеру страницы;
- По сравнению с размером адресного пространства, размер страницы очень мал.
- Накладные расходы при обращении к памяти: вначале к таблице страниц, а затем уже к памяти.
- Большой объем памяти, требуемый для хранения таблиц страниц.

Исходный код:

Для выделения памяти для страницы используется системный вызов **sbrk:** он расширяет область памяти программы и возвращает указатель на начало нового участка памяти, который и будет новой страницей. Полученный указатель записывается в массив указателей, увеличивая счетчик страниц.

```
void* alloc_page() {
   pages = (void**) realloc(pages, ++page_count);
   return pages[page_count - 1] = sbrk(page_size);
}
```

Проверка свободности указанного байта в памяти происходит проходом по массиву записей о выделенной памяти, который проверяет находится ли байт в рамках одного и выделенных участков.

```
bool is_free(void* addr) {
   auto addr_int = (intptr_t) addr;
   for (auto pair : allocated) {
        if (addr_int >= pair.first && addr_int <= pair.second) return false;
   }
   return true;
}
```

Поиск свободного места происходит путем линейного прохода по всей доступной памяти, этот алгоритм не очень эффективен и приводит к большим временным затратам на выделение памяти, когда уже выделено большое количество памяти.

```
bool find_free_space(intptr_t size, intptr_t& pg, intptr_t& ofst) {
   if (page_count == 0) {
       return false;
   }

   intptr_t page;
   intptr_t offset = -1;
   for (intptr_t i = 0; i < page_count; i++) {
       for (intptr_t j = 0; j < page_size - size + 1; j++) {
           bool found = true;
           for (intptr_t k = 0; k < size; k++) {
               if (!is_free((intptr_t)pages[i] + j + k)) {
                   found = false;
                   break;
               }
           }
           if (found) {
               offset = j;
               break;
           }
       }
       if (offset != -1) {
           page = i;
           break;
```

```
        }
    }

    if (offset == -1) {
        return false;
    }

    pg = page;
    ofst = offset;
    return true;
}
```

Выделение памяти использует вышеописанную функцию для нахождения свободного участка подходящего размера. В случае, если участок не найден, выделяется новая страница и запись произойдёт в неё с нулевым отступом от начала страницы.

Функция возвращает указатель на на найденный участок памяти и помечает этот участок как выделенный, чтобы предотвратить перезапись.

```
void* alloc(intptr_t size) {
    pthread_mutex_lock(&lock);
    intptr_t page = -1, offset = -1;
    bool succ = find_free_space(size, page, offset);
    if (!succ) {
        alloc_page();
        find_free_space(size, page, offset);
    }

    void* ptr = (void*)((intptr_t)pages[page] + offset);
    allocated[(intptr_t)pages[page] + offset] = (intptr_t)pages[page] + size +
offset - 1;
    pthread_mutex_unlock(&lock);
    return ptr;
}
```

Освобождение памяти происходит путем удаления упоминания о том, что память выделена. Данные остаются на месте, но могут быть перезаписаны при следующем выделении памяти.

```
void free(void* ptr) {
    auto it = allocated.find((intptr_t) ptr);
    if (it != allocated.end()) {
        allocated.erase(it);
    }
}
```

1. Чем ограничивается максимальный размер физической памяти, которую можно установить в компьютере определенной модели?

Максимальный размер физической памяти ограничивается разрядностью физической адресной шины.

2. Чем ограничивается максимальный размер виртуального адресного пространства, доступного приложению?

Максимальный размер виртуального адресного пространства ограничивается разрядностью адреса, присущего данной архитектуре компьютера.

3. В каких случаях транслятор создает объектный код программы не в виртуальных, а в физических адресах?

Транслятор создаёт объектный код программы в физических адресах тогда, когда заранее точно известно, в какой области оперативной памяти будет выполняться программа.

4. Что такое “свопинг”?

«Свопинг» - это подход к преодолению перегрузки памяти, который заключается в размещении полностью всего процесса в памяти, запуске этого процесса на некоторое определенное время, а затем его переносе на диск.

5. Как величина файла подкачки влияет на производительность системы?

Размер файла подкачки обычно должен быть равен объему оперативной памяти, слишком маленький или слишком большой размер файла подкачки может привести к проблемам с производительностью.

6. Почему размер страницы выбирается равным степени двойки? Можно ли принять такое ограничение для сегмента?

Размер страницы выбирается равным степени двойки для упрощения механизма преобразования адресов.

7. На что влияет размер страницы? Каковы преимущества и недостатки большого размера страницы?

Маленький размер страницы может привести к большому количеству обращений к ядру, что может негативно сказаться на производительности. Слишком большой размер страницы приводит к выделению большого количества неиспользуемой памяти.

8. Почему загрузка и выгрузка данных из кэш-памяти производится блоками?

Для того чтобы повысить объем полезной информации и одновременно уменьшить объем служебных данных. А вообще блоки это круто, потому что блоки есть еще и в лего, а лего - не просто крутая игрушка для детей, а игрушка с историей. Она, опередив «Монополию» и кукол Барби, призвана лучшей в мире. Это также объект

коллекционирования и увлечение многих взрослых. Оно и не удивительно, ведь само название конструктора, производное от «leg godt», переводится как «увлекательная игра».

9. Как обеспечивается согласование данных в кэше с помощью методов обратной и сквозной записи?

При сквозном методе запись выполняется одновременно и в кэш-памяти, и в ОЗУ. При осуществлении метода обратной записи цикл записи выполняется только в кэш-памяти.