**Министерство науки и высшего образования Российской Федерации**

федеральное государственное автономное образовательное учреждение высшего образования

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»**

**Дисциплина:**

«Операционные системы»

**Лабораторная работа №6**

“Malloc”

Выполнил:

Хоанг Вьет Зунг – Студент группы N32491

Нгуенг Хонг Ньят - Студент группы N32451

Преподаватель:

Савков С.В.



Санкт-Петербург 2023

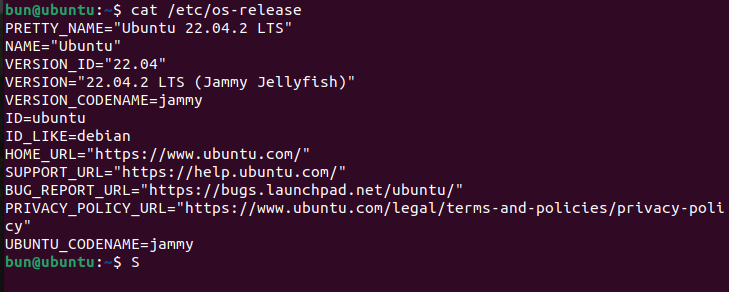
**Задание:**

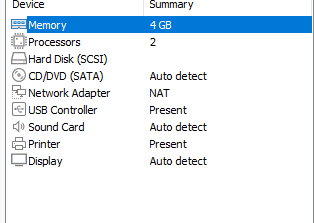
* Протестировать функцию malloc/free и построить график зависимости

времени выделения от размера запрашиваемой памяти.

Либо винда, либо линукс

* Сложный : Сравнить с другими малоками (calloc)

**Name: Ubuntu**

****

malloc() is used to allocate a block of memory of a specified size. It returns a pointer to the first byte of the allocated memory block, or NULL if the allocation fails. The malloc() function does not initialize the allocated memory, so the contents of the memory block are undefined until they are explicitly set by the program.

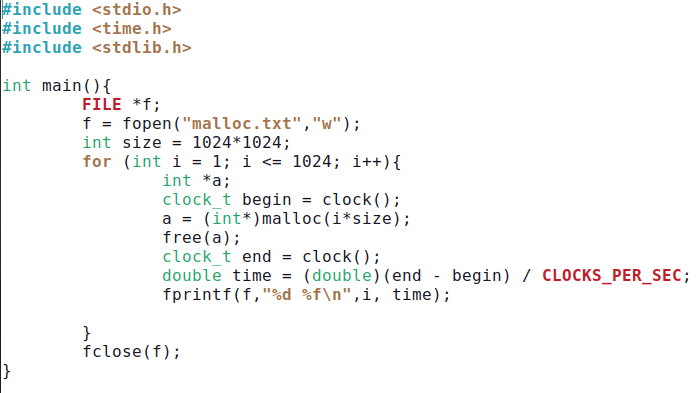
calloc() is similar to malloc(), but it also initializes the allocated memory to zero. It takes two arguments: the number of elements to allocate and the size of each element. The total size of the allocated memory block is the product of these two arguments.

realloc() is used to resize an existing block of memory. It takes two arguments: a pointer to the existing memory block, and the new size of the memory block. If the new size is smaller than the existing size, the excess memory is freed. If the new size is larger than the existing size, the function attempts to extend the existing memory block. If there is not enough contiguous memory available to extend the block, the function may have to allocate a new, larger block and copy the contents of the old block to the new block.

1. **Malloc():**

We used functions malloc() and free() in this program, measured the time taken to allocate and free memory and then wrote the result into file named malloc.txt. In this case, we allocated 1 MB in memory. The program then enters a loop that iterates from 1 to 1024, with the loop variable i representing the number of 1MB blocks of memory to be allocated.

Within the loop, the program first declares a pointer variable a of type int\*, which will be used to store the memory address returned by the malloc() function. The program then measures the time it takes to allocate memory using the malloc() function by calling the clock() function before and after the allocation, and calculating the elapsed time in seconds using the CLOCKS\_PER\_SEC constant. The elapsed time is stored in a variable named time. The program then deallocates the previously allocated memory using the free() function. The resulting "malloc.txt" file will contain two columns of numbers, with the first column representing the number of 1MB blocks of memory allocated and the second column representing the elapsed time in seconds for each allocation and deallocation operation.

****

**Chart, histogram

Description automatically generated**

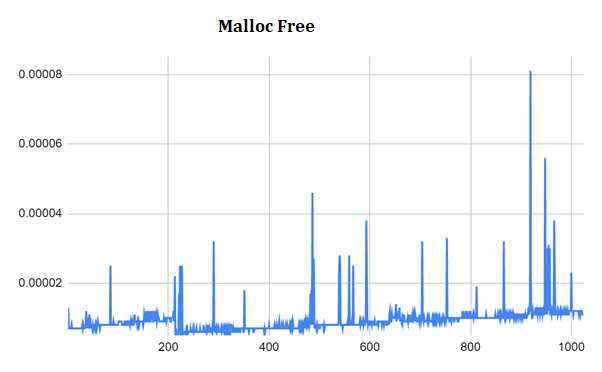
**Text

Description automatically generated**

The program is similar to the program above but in this case, we want to see the result in 2 separate files: allocation time and free time.

**Chart, histogram

Description automatically generated**

****

1. **Calloc()**

Text, letter

Description automatically generated

Chart, histogram

Description automatically generated

1. **Realloc()**

**Text

Description automatically generated**

**Chart, histogram

Description automatically generated**

**Table

Description automatically generated**

In general, malloc() is faster than calloc() because it does not initialize the allocated memory. Memory initialization can be a time-consuming operation, especially for large blocks of memory, so calloc() can be slower than malloc() for large allocations.

However, difference in performance between malloc() and calloc() may not be significant for small allocations. On the other hand, there are some cases where calloc() could be faster than malloc(). For example, if the program needs to allocate a large block of memory and then initialize it to zero, using calloc() could be faster than using malloc() followed by manual initialization. This is because calloc() can perform the initialization as part of the allocation process, potentially reducing the overall time required.

Realloc() can be faster than both calloc() and malloc() because it can avoid the overhead of copying memory when resizing an existing memory block.