Dr. Sam Wilson
School of Computing
University of Leeds

Coursework 1

Operating Systems (COMP2211)

You should provide a solution to the problem outlined below. You may choose to submit a report rather than source code. If you choose to submit a report you will be unable to achieve a grade higher than 70%.

Submission You **must** submit a copy of your code/report via two channels.

1. **Minerva:** You should compress your source code into a **.tar.gz** file and submit via the appropriate Minerva submission point. For submitting the report you should upload a **.pdf** file to the appropriate Minerva submission point. **No other file formats will be accepted.**
2. **Gitlab (code only):** You should use gitlab to version control your source. You should use the repository provided by the module leader. You should commit little and often.

Deadline: TBC

Award: This piece of summative coursework is worth 10% of your grade.

Memory Management

When programming in the C programming language dynamic memory allocation is facilitated by calls to `malloc`, `free`, `realloc` and `calloc`. "Under the hood" these calls are dynamically expanding and shrinking the heap memory segment of the process.

Your task is to implement functions similar to `malloc` and `free` such that they meet the following specification.

or

Your task is to write a technical report which describes the process of memory management that `malloc` and `free` implement. The report should be detailed enough such that a competent C programmer could implement `malloc` and `free` from your description. The technical report should be no longer that 4 sides of A4 (not including references); you should use diagrams where appropriate. **Please be aware that the maximum grade you can achieve by submitting a report is 70%.**

malloc

The `malloc` function allocates the specified number of bytes and returns a pointer to the allocated memory. The memory is not initialized. If the requested size is 0, then `malloc()` returns either `NULL`, or a unique pointer value that can later be successfully passed to `free`.

- the function should have the prototype `void _malloc(size_t size)`
- the function should return a void pointer to the start of the allocated memory
- the function should allocate memory on the heap
- the pointer returned should be suitably aligned for any kind of variable (advanced)

free

The `free` function frees the memory space pointed to by the parameter which must have been returned by a previous call to `_malloc()`. Otherwise, or if `free` has already been called on the pointer before, undefined behavior occurs. If the pointer is `NULL`, no operation is performed.

- the function should have the prototype `void _free(void *ptr)`
- the function should "free" the memory and make it available for subsequent allocations
- the function should return the memory to the operating system when it is appropriate (advanced)

Report

You can choose if you wish to implement the coursework in the C programming language or whether you write a technical report outlining how the memory management works in detail, including any auxillary data structures that are used.

Marking scheme

Marks will be allocated equally for implementing/describing `malloc` and `free`. Marks will be allocated for the quality and consistency of your code or the clarity of your technical description.

Academic misconduct

The University's academic misconduct standards apply. There are many resources online that could support you in completing this coursework. There are no problems in reading around in order to gather a better understanding of different implementations of memory management function in the C programming language. However, **all** sources of information should be appropriately referenced and proper attribution should be provided. The work that you submit should be **entirely your own work**, written by you.

The module staff will pass on any suspected misconduct to the School of Computing's misconduct officer.