Operating Systems (Abstract)

Filip Schauer, 4AHITN 2020/21

The research paper briefly covers a few key areas of low-level functionality of operating systems, with a focus on the x86 architecture and general-purpose operating systems.

As part of the research, forums, wiki pages, open-source projects and datasheets were studied and an own x86 general purpose operating system was developed from scratch. For the comparison of the task scheduling algorithms, the x86 operating system written by the author was used and the individual algorithms were implemented one after the other. Each algorithm was tested with different loads 510 times in a virtual machine on an AMD Ryzen 5 1600 processor. Profiling information was written during runtime by the kernel in an efficient way to minimize the influence on the results. The profiling information of all runs was analysed using a program written by the author. This program created a Gantt chart for each run and a CSV file that was later imported into Microsoft Excel to create graphs.

How does the kernel get loaded into memory and then get executed?

The firmware first loads and runs the bootloader. The bootloader then loads the kernel into memory, sets up a suitable environment and transfers control to the kernel.

How can an operating system ensure that processes do not get in each other's way by writing to the other's allocated memory area?

Modern processors have a Memory Management Unit (MMU) which allows the system to restrict memory access for different processes to only their own allocated memory areas. The commonly used scheme for this is called Paging and involves mapping virtual addresses to physical addresses. Each process is given its own virtual address space through which it can only access its own memory.

Which process scheduling algorithm performs best?

Of those tested, Multilevel Feedback Queue was the clear winner. Although it is more complex to implement than some other algorithms, interactive tasks achieve very good response times and CPU-heavy tasks have a chance to recover their position in a higher priority queue if they behave well.