Kyle's Notes and Summary of "THE"

Paper Read: [1]

1 Notes

Multiprogramming: the system supports concurrent processes. Problem: how do we design, build, and verify the correctness of a complex system with limited resources? The solution is effectively:

1. Minimize typical ideas that impose drudgery
2. Use sane hardware
3. Critical retrospectives

"THE" is effectively a case study in this problem and solution. With "THE", the system needed to handle a steady input stream of programs, operating in batch. "THE" was specifically optimized for short-running, I/O bound, ALGOL 60 programs. Designed it to allow for verifying its correctness ahead of time and to enable exhaustive testing; this method of design is considered the main contribution.

Introduced a primitive form of virtual memory; programs would operate in abstract segments, and segments would map to either core or drum pages. "THE" took care of recording where the segment was at any point, transparent to the user program. Depending on whether or not swapping occurred, the same segment may end up at different physical locations of core memory at different times.

Programs are executed sequentially in a manner that is not dependent on wall clock time (i.e., real time vs. CPU time). Each input and each output is mapped to programs. This all requires cooperative program synchronization for any sort of communication between processes.

System obeys a strict hierarchy:

- **Level 0**: process scheduling. Allows preemption, and will not schedule blocked processes. Likely lives in core memory.
- **Level 1**: virtual memory system. Handles paging as needed. Likely lives in core memory. Levels above talk in segments.
- **Level 2**: keyboard and command interpreter. Needed in order to forward commands to appropriate programs; the user needs to say which program she is addressing. Intentionally put above level 1 so it can be paged out. Levels above operate as if they are speaking directly to the user.
- **Level 3**: I/O buffering/unbuffering.
- **Level 4**: User programs.
- **Level 5**: Operator

The system was developed incrementally according to this hierarchy, starting from the bottom and working up. Implies that testing was done on each level before moving up, but it also states that testing is not complete (perhaps some testing was done, and then more later?). The tests themselves were complex, and ensuring that all program states were exhaustively explored was challenging. Fewer levels would have caused an exponential state explosion during testing.

Appendix concerns semaphores, which are the basic synchronizing primmitive used to coordinate processes. P operation decrements; blocks if result is negative. V operation increments; can unblock a process if the result is non-positive. Proves that processes can only block when they progress forward, and processes can unblock if other processes progress forward. As such, the system as a whole progresses (assuming no deadlock; this is handled later by basically saying "we code in this particular way, and this cannot lead to deadlock").

2 Summary

Dijkstra seeks to solve the problem of designing, building, and verifying major software systems, using only limited hardware and developer resources. Dijkstra's solution is composed of three parts: critical retrospectives of past

design and development decisions, minimization of typical ideas that impose drudgery, and the use of sane hardware. As a case study to demonstrate the validity of this solution, Dijkstra and his team implement the "THE" multiprogramming system, a batch-oriented operating system specifically optimized for short-running, I/O bound, ALGOL 60 programs. True to Dijkstra's three-part solution, past mistakes informed system design, the system utilized a radical layered implementation approach, and the system was implemented on well-designed hardware. Each layer consisted of a program which communicated with programs on lower levels in a strict hierarchical fashion, enabling incremental system implementation, easy reasoning about each layer, and exhaustive testing of layers in relative isolation. This paper also introduces concurrency, semaphores, and paging. While "THE" is a successful case study demonstrating Dijkstra's solution, it is but one case study, and its evaluation is qualitative and anecdotal.

References

[1] Edsger W. Dijkstra. The structure of the the-multiprogramming system. *Commun. ACM*, 11(5):341–346, May 1968.