ADJ Assignment 1
SER 334: Operating Systems

Swanson, Douglas
dswanso6@asu.edu

Speidel, Clay
cspeidel@asu.edu

November 27, 2019

1 Question 1: Operating System Structures

1.1 Problem (P)

[Acuña, Lisonbee] Different operating system structures offer both benefits and drawbacks over one another, and it's important to understand what kind of structure should be used for different use-cases. Consider the following situation. You are designing an operating system for an interstellar probe that is meant to run for hundreds of years. The probe should also support monitoring many specific instruments.

What structure (simple, layered, microkernel, or modular) should you choose for its kernel?

1.2 Analysis (A)

An interstellar probe would have some of the following requirements on it's operating system kernel. The operating system would need to be robust, self reliant, memory efficient, power efficient, and support various sensors and subsystem equipment. It should also allow the programmers to develop the kernel almost independently from the spacecraft and allow for extensive testing.

The kernel will need to be robust means that it would handle its own errors and anomlous conditions in a manner that would allow it to continue operating normally thus keeping the satellite alive for the entirety of its mission. This should handle things like subsystem failures. This requirement compliments the self reliant requirement becuase to be self reliant it should be able to make decisions, that are in the best interest of the spacecraft's mission, in solving the errors and anomlous conditions that may appear over it's lifetime.

This kernel should be both memory and power efficient because a spacecraft has very limited tolerances for both of these resources. The memory should be conserved because the length of the mission is so very long, hundreds of years, thus the limited memory of the system would need to last. The power is another finite resource which will, over the lifetime of the mission, become reduced, thus the kernel should require minimal power.

The spacecraft will have multiple subsystems on it, which will range from core function such as temperature, attitude, and navigational control to more mission specific functions such as data logging and payload control. The kernel will need to allow for the addition of these modules as well as handle their health and data.

The requirements stated above become the following technical requirements:

- Handle all of the iteractions between the software and the hardware.
- Allow for scheduling of tasks over long periods of time.
- Control interactions between subsystems, not allowing one subsystem to diretly control another.
- Allow for new or different subsystems to be added during development.

The first requirement would allow for the kernel to have direct and exclusive control over the hardware which would promote the robustness and self reliance of the design because it would only allow processes that have been approved and tested to act upon the hardware. This reduces the risk of anomlous hardware condtions, only our processes will be used, as well as promotes easy testing because only the prescribed processes would need to be tested.

The second requirement would assist with the memory and power requirement as well as supporting inter subsystem timed communications. This would allow the system to to toggle subsystems on and off when they are required to do something thus reducing the power and memory requirements to the bare minimum at any given time.

The third requirement allows for the operating system on the spacecraft to have complete control over all of the subsystems onboard which will allow it to have complete information of all of the messages generated on board as well as the state of each of the subsystems. This gives the operating system the ability to make informed decisions to maintain the health and sucess of the mission.

The final requirement decreases the development time of the operating system which will decrease the overall labor cost on the spacecraft. This also decreases both the risk of failure of the operating system as well as the amount fo time spent testing. The testing is reduced because everytime a subsystem on the spacecraft is added, removed, or changed the entirety of the operating system doesn't need to be retested which in turn reduces the changes to the system and the percentage of human errors appearing in the code.

1.3 Design (D)

There are two kernel structure that was choosen was the microkernel. This structure satisfies the requirements listed above in the following ways. The microkernel, by definition, handles all communication between the plugins attached to its core. By controlling the communication between the subsystem plugins the core can then authenticate the requests between subsystems before passing them along which would allow it to catch bad requests before they are sent to the subsystem, thus reducing the risk of a subsystem breaking another subsystem. This will also allow the core to track all of the communications on the spacecraft which could be useful should the system enter an anomlous state that requires debugging. This core of the system will aslo handle accessing the hardware and scheduling of tasks. Since the micro kernel core can monitor all of the modules on board, it can then make informed decisions and safe guard the rest of the spacecraft in emergency situations. The microkernel would have an API that would allow the team to pick and choose their sensor and subsystems through the development process and write moduluar code that interacts with the core in a predefined manner, thus making testing of modules and the core easier. Also allows for flexibility throughout the development cycle without having to redesign the core kernel.

1.4 Justification (J)

The design stated above out performs the other operating system structures and is the optimal solution. Take a look at the simple structure, it would allow for everything on the spacecraft to directly access the spacecraft hardware which would be very bad if one of the subsystems failed for some reason and was commanding the spacecraft hardware to do things based upon erroneous data that could cause the mission to fail. The layered approach would solve this issue but it could be harder to maintain throughout the development of the spacecraft, harder to test, because when a new, or different, subsystem is added to the spacecraft it would have to be integrated into a layer in the kernel. Everytime we change a layer the entire kernel would have to be retested. Finally the modular structure would bring all of the benefits of the microkernel but would allow the subsystems to talk directly to each other. This cross communication would be a benefit on a system that was being actively monitored and accessed by a human, but on our spacecraft we would want our core to have complete control and knowledge of the entire system so that it can perform emergency actions. The cross communication also introduces the risk of a lazy developer bypassing the core and having their subsystem directly command another, which becomes a problem should their subsystem break and start commanding another subsystem to do things based upon erroneous data. So in the end the microkernel process would be the best structure to use.

2 Question 2: Processes

2.1 Problem (P)

2.2 Analysis (A)

2.3 Design (D)

2.4 Justification (J)

3 Question 3: Threads

3.1 Problem (P)

3.2 Analysis (A)

3.3 Design (D)

3.4 Justification (J)