1.

System calls provide an interface to the services made available by an operating system5.
System calls can be grouped roughly into six major categories: **process control**, **file management**, **device management**, **information maintenance**, **communications**, and **protection**

Process control5

- create process, terminate process
- load, execute
- get process attributes, set process attributes
- wait event, signal event
- allocate and free memory

- **File management5**
  - create file, delete file
  - open, close
  - read, write, reposition
  - get file attributes, set file attributes
- **Device management5**
  - request device, release device
  - read, write, reposition
  - get device attributes, set device attributes
  - logically attach or detach devices
- **Information maintenance5**
  - get time or date, set time or date
  - get system data, set system data
  - get process, file, or device attributes
  - set process, file, or device attributes
- **Communications5**
  - create, delete communication connection
  - send, receive messages
  - transfer status information
  - attach or detach remote devices
- **Protection5**
  - get file permissions
  - set file permissions

2.

الف) به دلیل نیاز به سرعت بالا از shared memory استفاده می شود.3

7یک نمونه نحوه ارتباط: ابتدا یک فضا به صورت shared تعریف می شود. به اندازه یک عدد Int برای شمارنده مقدار پیام خوانده نشده فضا درنظر می گیریم.

Counter : تعداد پیام های باقی مانده در فضای مشترک

Limit: تعداد پیام هایی که می توان ارسال کرد و پس از آن باید صبر کنیم تا پیام ها دریافت شوند.

هر بار که فرآیند اول می خواهد پیامی بفرستد counter را می خواند. اگر برابر صفر بود پیام ها را می فرستد و counter را برابر تعداد پیام ها می گذارد. اگر صفر نبود زمان t را صبر کرده و سپس دوباره مقدار counter را چک می کنئ.

فرآیند دوم برای دریافت پیام ها،ابتدا مقدا counter را چک می کند،اگر مخالف صفر باشد هرپیامی را که می خواند یک واحد مقدار counter را کم می کند. همچنین اگر با مقدار صفر مواجه شود پس از t ثانیه دوباره counter را چک می کند.

ب)

3در روش message passing پیش از ارسال پیام یک ارتباط باید برقرار شود و می توانیم در این حین اطلاعات مربوط به مسئله امنیت را بررسی کنیم درنتیجه از این تکنیک استفاده می کنیم.

7یک نمونه نحوه ارتباط:

1. ایجاد یک ارتباط و بررسی اطلاعات مورد نظر مانند ip , pid
2. ارسال با استفاده از send(x)، از آنجا که طول پیام ها متغیر است تابع send را به گونه ای پیاده سازی می کنیم که در ابتدا طول پیام را ارسال کند، یعنی x= message size سپس بلافاصله به ارسال پیام بپردازدx = message content.(می توانیم پیام را به قطعات برابر بشکنیم یا هرتکنیک دیگری)
3. دریافت با استفاده ازreceive() ابتدا مقداری را که دریافت می کنیم برابر طول پیام درنظر گرفته، سپس منتظر دریافت خود پیام می مانیم.

3.

الف) در یک سیستم می خواهیم به این سوال که کدام نخ باید شانس اجرا را در نوبت بعد داشته باشد پاسخ دهیم برای مثال آیا نخ ای که زودتر ایجاد شده است یا نخ ای که اولویت بالاتر دارد باید در نوبت بعد اجرا شود؟2 policy

برای این منظور سیستم تعویض بین نخ ها را پیاده سازی می کنیم .2 mechanism

ب) یک سایت نیازمند این است که یک سیستم برای ورود کاربران خود داشته باشد policy2

کاربران می توانند از طریق نام کاربری و رمز عبور وارد شوند1 Mechanism .
همچنین آنها می توانند از ایمیل خود نیز برای ورود استفاده کنند2 mechanism
2+2

4.

- Benefits:
  - Easier to extend a microkernel
  - Easier to port the operating system to new architectures
  - More reliable (less code is running in kernel mode)
  - More secure
- Detriments:
  - Performance overhead of user space to kernel space communication

---

5.

یک نمونه :

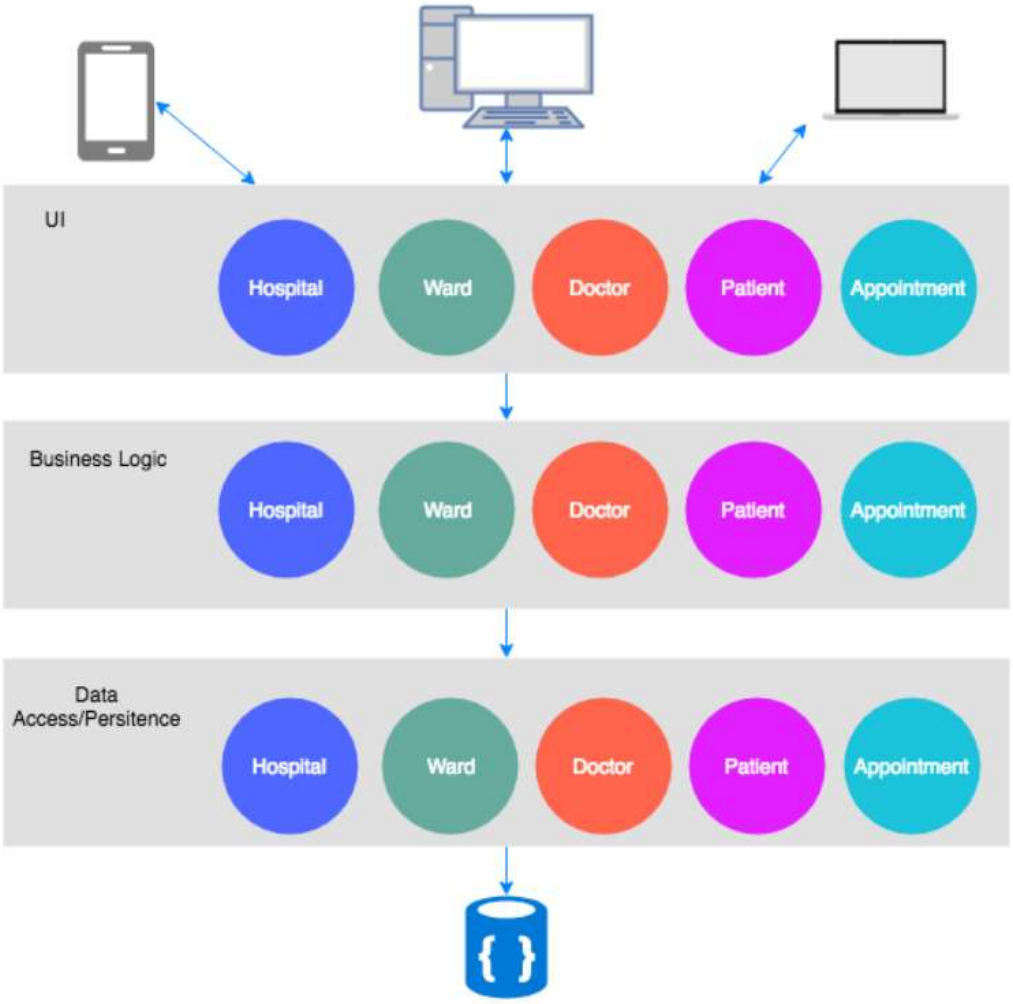

Figure 2 — Monolithic Layered Architecture of a medical system.

Layered architecture is a common pattern seen in monolithic applications. This may be due to the initial structure or lack-there-of it, and it's suitability to the problem at hand

This architecture allows for the technical capability to be changed fairly easily, especially if they are isolated to a particular layer. The use of interfaces for the purpose of isolating layer dependencies makes sense. It also makes sense when it comes to testing because small units are easily testable if confined to the boundaries of each layer. Changing the database technology for a different one is an example of a change in technical capability which implies changes

to only one of the layers. We could also change the UI framework from one to another. Technical changes are only one side of the problem; a change in business requirement implies a change across all layers, and that means we have to coordinate those changes across each layer. Imagine there is a new business requirement — *A patient must be able to specify if they'd like to be sent a reminder when creating an appointment*. As we can see in figure 2, the **Appointment** domain object is represented across all the layers, so we must change all these layers even though only one part of the domain has changed.