# Fragebogen 5 – Interprozess-kommunikation

1. Welche drei prinzipiellen Arten der Interprozess-Kommunikation werden in der Praxis benutzt? Schildern Sie kurz deren Funktionsweise.
   1. Speicherbasiert
   2. kleiner Bereich in den Prozessen ist VM (Virtueller per MMU realisierter Adressraum) bzw. Shared Memory
2. Zwei Threads in verschiedenen Prozessen auf einem System wollen miteinander kommunizieren. Welche der drei Arten von Interprozess-Kommunikation ist am schnellsten?
   1. Shared Memory durch VM (Virtueller per MMU realisierter Adressraum) je nach OS
3. Zwei Threads im selben Prozess wollen miteinander kommunizieren. Welche der drei Arten von Interprozess-Kommunikation wird immer angewendet?
   1. Shared Memory/gemeinsam benutzter Speicher
4. Mit Hilfe welcher Hardware-Komponente wird die Interprozess-Kommunikation mit der Benutzung gemeinsamen RAMs zwischen Prozessen realisiert (Virtueller Speicher wird vorausgesetzt)?
   1. Durch die Memory Management Unit (MMU).
5. Zwei Threads innerhalb eines Prozesses kommunizieren durch Austausch von Nachrichten. Müssen diese kopiert werden? Wenn die beiden Threads unterschiedlichen Prozessen angehören: müssen diese kopiert werden?
   1. Müssen nicht kopiert werden, werden aber häufig kopiert. Nur bei Shared Memory nicht, ansonsten immer
6. Für welche Aufgaben lassen sich zirkuläre Puffer gut verwenden?
   1. Zur Kommunikation zwischen Tasks.
7. Skizzieren Sie die Idee der zirkulären Puffer. Wie verläuft das Einbringen einer Nachricht (enqueue) und wie das Herausholen (dequeue)?
   1. enqueue - Inhalt des Index in wird beschrieben. Mithilfe Modulo wird geschaut, ob das Ende erreicht ist.
   2. dequeue - Inhalt des Index out wird geholt. Mithilfe Modulo wird geschaut, ob das Ende erreicht ist
8. Warum werden die zirkulären Puffer als zirkulär bezeichnet?
   1. Weil die Füllzeiger in und out immer im Array im Kreis herumlaufen. Hinterste Element ist verbunden mit dem ersten Element.
9. Was wird unter den Producer-Consumer-Problemen verstanden? Was ist dabei das Problem?
   1. Producer will schreiben wenn voll
   2. Consumer will lesen wenn leer
   3. Beide greifen auf einen gemeinsamen Buffer zu.
   4. Derjenige der produziert muss auch platz zum abspeichern haben . Consumer kann nicht mehr consumen als er Kapazität hat
10. Beschreiben Sie die Funktion und prinzipielle Implementierung von Pipelines (Pipes) in UNIX-Systemen. Welche Komponente implementiert diese Pipelines?
    1. Eine Queue vom Betriebssystem erstellt und verwaltet
    2. Nutzung von pipe-Syscall
    3. Das Betriebssystem
11. Worin besteht der wesentliche Unterschied zwischen Pipes und Named Pipes in UNIX? Für welche Art von Anwendungen lassen sich Named Pipes gut verwenden?
12. Wenn ein Prozess einen Port hat, was ist damit gemeint? Also, was ist ein Port und worin besteht der Unterschied zu einer Named Pipe?
    1. Port: Adresse mit symbolischen Namen. Feste Verbindung zwischen Server und Schnittstelle. Server müssen Semantik erbringen.
    2. Named Pipe: = Pipeline . Keine feste Verbindung zwischen Server und Schnittstelle. Beliebiger Server kann auslesen und irgendwie verarbeiten.
13. Wenn alle Prozesse (und auch Threads) Ports haben, gibt es dann noch einen wesentlichen Unterschied zwischen Clients und Servern? Wann ist eine Task dann ein Server und wann ein Client?
    1. Server und Client sind nur Rollen. Diese Rollen kann jeder einnehmen.
    2. Clients: nutzt sendOperationen
    3. Server: nutzt receiveOperation
14. Bei der Interprozess-Kommunikation wird zwischen synchron und asynchron unterschieden. Wann ist ein send() synchron und wann asynchron?
    1. asynchron: Sender und Empfänger zeitlich entkoppelt. Empfang nicht quittiert.
    2. synchron: Das send() wartet auf Bestätigung der Verarbeitung.
15. Was wird unter dem Rendezvous-Verfahren verstanden?
    1. Eine synchrone Kommunikation mit reply() Operation mit Bezugnahme auf die jeweilige Nachricht (eindeutige Nummer).
16. Worin besteht der wesentliche Unterschied zwischen einem synchronen send() und einem blockierenden send()?
    1. Blockieren - Fortschritt innerhalb Operation
    2. Synchron - Verhältnis zweier Partner
17. Beschreiben Sie die Realisierung eines asynchronen send(), wenn nur synchrone send()-Operationen zur Verfügung stehen.
    1. Zwischenspeicher (Queue)
    2. zweiter Sender, der stellvertretend kommuniziert.
    3. wie Just-in-time
18. Wann sind ganz allgemein Queues bei der Kommunikation erforderlich?
    1. häufige asynchrone Kommunikationunterschiedliche Geschwindigkeiten bei Beteiligten
19. Unter welchen Bedingungen kann eine Sende-Operation einer Nachricht blockieren? Nennen Sie zwei.
20. Kann eine wartende Empfang-Operation scheitern? Konstruieren Sie dazu eine pathologische Situation.
21. Beschreiben Sie die Idee des Publish-Subscribe-Musters bei Nachrichten-orientierten Systemen.
22. Was bedeutet es, wenn eine Task bei einem Broker ein Topic anmeldet?
23. Nennen Sie drei Aufgaben des Brokers bei Publish-Subscribe-Verfahren?

# Fragebogen 6 – Remote Procedure Call Nr. 1 (RPC)

1. Definieren Sie den Remote Procedure Call als Gegensatz zum Local Procedure Call.
   1. LPC - Lokaler Prozeduraufruf innerhalb eines virtuellen Speichers (VM)
   2. RPC - Entfernter Prozeduraufruf (Routine liegt in einem anderen virtuellen Speicher / Host als der Aufrufer)
2. Können zwei Threads mit einem Remote Procedure Call miteinander kommunizieren? Unter welcher Bedingung?
   1. Ja, wenn nicht im selben Prozess.
3. Mit welchen besonderen Problemen muss sich die Implementierung des Remote Procedure Calls im Gegensatz zum lokalen Prozeduraufruf beschäftigen? Nennen Sie zwei.
   1. hohe Wartezeiten aufgrund von unterschiedlichen Geschwindigkeiten
   2. Deadlock durch Paketverlust
   3. Dubletten (mehrfacher Datensatz) beim Empfänger aufgrund von zu langem Paketversand
4. Warum kann beim Remote Procedure Call in jedem Falle vom Client und vom Server gesprochen werden? Was ist in dieser Hinsicht anders als beim einfachen Nachrichtenaustausch zwischen Gleichberechtigten?
   1. Aufgrund der Semantik per Definition.  
      Client - Routine aufrufen  
      Server - Routine realisieren
   2. Wenn Server B fehlerhaftes produziert, dann ist auch Client A fehlerhaft
5. Was macht die reply()-Operation anders als die send()-Operation auf der Seite des Servers?
   1. Eine synchrone Kommunikation mit reply() statt send() Operation mit Bezugnahme auf die jeweilige Nachricht (eindeutige Nummer).
6. Welche Möglichkeiten der Reaktionen bestehen, wenn der Verlust eines send()-Requests auftritt? Welche vier Arten von RPC-Semantik können prinzipiell daher realisiert werden?
   1. Senden kann wiederholt werden. - Maybe - At-least-once - At-most-once - Exactly-once
7. Die Möglichkeit von Dubletten von Datenpaketen kann dazu führen, dass beim RPC modifizierende Operationen doppelt ausgeführt werden. Wie lässt sich dies verhindern?
   1. Durch eine Prüfsumme
8. Aufgrund von Ausfällen von Systemen kann es beim RPC zu Waisen kommen. Was ist ein Waise? Und wie entsteht ein solcher?
   1. Eine Server-Aktivität, dessen Auftraggeber nicht existiert.
   2. Ein Waise kann beim Ausfall einer Komponente entstehen
9. Beschreiben Sie ein Verfahren zur Waisenbehandlung mittels Epochen. Was ist hierbei eine Epoche?
   1. Mit jedem Start des Client beginnt eine neue Epoche mit einer bestimmten ID, die den Paketen/Aufträgen mitgegeben wird, so dass veraltete Pakete identifiziert werden können.
   2. Epoche - Zeitabschnitt mit sequentiell nummerierten Teilabschnitten
10. Wenn ein Zusammenbruch einer verteilten Anwendung festgestellt wird, müssen eventuell schon erfolgreich durchgeführte Modifikationen zurückgenommen werden. Begründen Sie dies anhand eines Beispiels.
    1. Inkonsistenz kann entstehen. ABC-Beispiel: A & B haben Zustand aktualisiert, C aber nicht.
    2. Falls ein Aufruf in einem Kontext erfolgt, sollten die Modifikationen zurückgenommen werden. Bsp.: Transaktionen.
11. Mit den DB-Transaktionen können alle Effekte mittels rollback() zurückgenommen werden. Was ist erforderlich, wenn ein commit() schon durchgeführt wurde und die Effekte anschließend zurückgenommen werden sollen?
    1. Kompensationstransaktion, indem alle Operationen einzeln zurückgenommen werden.
12. Beschreiben Sie die beiden Phasen des 2PC-Verfahrens zur Realisierung von Transaktionen in verteilten Anwendungen.
13. Beim 2PC-Verfahren zur Realisierung von Transaktionen fällt der Koordinator aus und es gehen sämtliche Informationen über die Transaktionen dadurch verloren. Kann dann immer noch die ACID-bedingung eingehalten werden? Und falls ja, wie?
14. Was passiert beim Write-Ahead-Logging-Verfahren (WAL)?

# Fragebogen 7 – RPC Nr. 2

1. Welche Aufgaben hat der Stub beim Remote Procedure Call? Nennen Sie mindestens zwei.
   1. Serialisieren und Deserialisieren.
2. Stubs werden zum Remote Procedure Call benötigt. Benötigt der Austausch von Nachrichten auch Stubs?
   1. Natürlich, selbstverständlich.
3. Aus welchen Gründen müssen die Parameter des Remote Procedure Calls auf der Senderseite serialisiert und auf der Empfängerseite deserialisiert werden?
   1. Senden von komplexen Datenstrukturen - Semantisch gleiche Repräsentation
4. Gilt der Zwang zur Serialisierung/Deserialisierung auch für die Resultate von Funktionen?
   1. Ja
5. Wenn mit einem Remote Procedure Call eine Funktion aufgerufen wird: was muss alles beim Rücktransport der Informationen zum Aufrufer übertragen werden? (Es sind zwei Arten von Informationen)
   1. Liste der Resultate (Informationen) - ExceptionInformationen (MetaInformationen)
6. Wenn verkettete Strukturen, z.B. Objekte vom Typ List<> in Java, serialisiert werden, muss ein bestimmtes nicht einfach zu lösendes Problem gelöst werden, welches?
   1. Datenstruktur exakt an der Stelle liegen, an der sie beim Client liegt
   2. Referenzvariablen zeigen vermutlich falsch.
7. Wie können Referenzen mit Pointern serialisiert werden? Beschreiben Sie drei Vorgehensweisen und bestimmen Sie deren Vor- und Nachteile.
   1. Verbieten (Nachteil: wird häufig gebraucht)
   2. Referenzobjekte mit serialisieren (Nachteil: Ineffizient)
   3. Referenzobjekte lokal verfügbar machen. Objekte zwischen Knoten verschiebbar. (Nachteil: Management von Speicherorten)
8. Wie sieht die grundsätzliche Software-Struktur beim RPC auf der Seite des Servers aus?
   1. Outline bestehend aus einer Endlosschleife. - Pakete empfangen - Namen auslesen - Prüfen, ob Name auf dem Server existiert - Parameterliste entpacken - Routine aufrufen - Ergebnis serialisieren - Nachricht zurückschicken
9. Eine Routine soll vom Aufruf zum nächsten mit unterschiedlich vielen Parametern aufgerufen werden können. Welches besondere Problem muss dazu gelöst werden?
10. Eine Routine soll vom Aufruf zum nächsten mit unterschiedlich vielen Parametern an derselben Stelle im Code aufgerufen werden. Welches Problem muss die aufrufende Routine dazu lösen? Warum besteht dieses Problem nicht, wenn die Aufrufe mit unterschiedlich vielen Parametern an verschiedenen Stellen im Code gemacht werden?
    1. Die Routine muss unterschiedlich viele Parameter handlen können. Weil überladen werden kann
11. Was wird unter Reflexion bei Programmiersprachen verstanden? Über was wird reflektiert und was ist dessen Ergebnis?
    1. Eigene Programmstruktur selbst analysieren. Über die Symboltabelle des Compilers.
12. Wie wird bei Java und anderen per Compiler übersetzten Sprachen Reflexion prinzipiell zur Laufzeit realisiert?
    1. Zur Laufzeit liegen Symboltabellen vor
13. Wie wird bei Skript-Sprachen wie z.B. JavaScript oder PHP Reflexion prinzipiell intern im Interpreter realisiert?
    1. Bei SkriptSprachen ist sämtlicher Quelltext vorhanden.
14. Was bedeutet technisch, wenn von einem nicht-wartenden send() oder receive() gesprochen wird? Welche Konsequenzen hat das „Nicht-Warten“?
    1. Es können zwei Befehle bei einer Komponente gleichzeitig ankommen.
15. Welche Reaktionen sind sinnvoll, wenn beim nicht-wartenden send() festgestellt wird, dass die erforderliche Sende-Queue im eigenen System voll ist und daher das send() nicht ausgeführt werden kann?
    1. send () muss Trotzdem Warten Es wird auf freie Plätze gewartet.
16. Auf dem Server sollen möglichst parallel viele RPCs bzw. Nachrichten verarbeitet werden. Wie sieht eine dafür geeignete Software-Struktur aus?
    1. Scheduler - Queue mit Prioritäten - Danach wird weiterverteilt
17. Treten bei wartenden (klassischen) RPCs Probleme des wechselseitigen Ausschlusses innerhalb einer verteilten Anwendung auf?
    1. Nein, da die Aufrufe nicht parallel ablaufen.
18. Was muss bei der Benutzung mehrfacher asynchroner send()-Operationen innerhalb einer verteilten Anwendung beachtet werden?
    1. Es entsteht ein kritischer Abschnitt
19. Beim Starten eines verteilten Systems müssen die beteiligten Komponenten erzeugt werden. Wie kann dies realisiert werden? Skizzieren Sie eine prinzipielle Lösung.
    1. Es wird eine anwendungslokale Instanz der betroffenen Komponente gebildet.
    2. Eine Anwendung besteht aus mehreren Instanzen von Komponenten auf mehreren Knoten
20. Wenn unter einer Komponente das Äquivalent einer Klasse verstanden wird, was wäre dann bei den Komponenten eine Instanz?
    1. Objekt
21. Warum sollten alle laufenden Anwendungen jeweils eine eindeutige ID besitzen? Was kann anhand dieser ID bestimmt werden?
    1. Damit nicht alle Instanzen gemeinsam genutzt werden. Die ID bestimmt welche Instanz einer Komponente zu welcher Anwendungsinstanz gehört.
22. Auf einem Server liegen z.B. zwei Instanzen derselben Komponente. Nun wird eine Nachricht empfangen und muss den Instanzen zugeordnet werden. Beschreiben Sie dazu den prinzipiellen Algorithmus.
    1. ??? Sofern IDs vorhanden sind, werden die Instanzen anhand dieser zugeordnet
23. Wie kommen in einem verteilten System Deadlocks zustande? Konstruieren Sie dazu ein Beispiel.
    1. Rekursion bzw. gegenseitiges Warten
24. Nennen Sie mindestens zwei Gründe, warum jeder RPC-Aufruf eine eindeutige Nummer haben sollte.
    1. Mehrere nichtwartende send() ausführbar
    2. Ergebnisse lassen sich leicht zuordnen

Fragebogen 8 Namensdienste & Namens-Pepository

1. Bei den Namensdiensten wird zwischen Adressen und Namen unterschieden. Worin liegen die Unterschiede? Definieren Sie beide.
   1. Name - Ortsunabhängige, symbolische Identifikation (Websiteadressen, Email-Adressen)
   2. Adresse - Identifikation eines Ortes oder eimes Objektes an diesem Ort (z.B. IP oder RAM)
2. Was unterscheidet einen Namen für einen Dienst von einem Namen von einem Server?
   1. Dienstname ist die Abstraktion von Namen von Servern.
3. Welche Aufgaben hat ein Namensdienst bzw. Verzeichnisdienst (Name Service, Directory Service)? Nennen Sie dazu drei typische Operationen dieses Dienstes.
   1. Allgemeine Informationen über Objekte, Personen oder Organisationen bereitstellen
4. In der Datenbank eines Namensdienstes werden verschiedene Informationen gespeichert. Nennen Sie dafür drei Beispiele.
   1. - Adresse - Name - Art des Dienstes – Kosten
5. Wenn zwei Klienten sich innerhalb eines Netzwerks finden wollen: was müssen sie dazu tun, wenn sie ohne Broadcast-Operationen auskommen wollen?
   1. Sich beim directory Server anmelden
6. Wofür steht das Kürzel DNS? Was ist die wichtigste Aufgabe des DNS-Dienstes?
   1. Domain Name Services
   2. Umsetzung symbolische Adresse nach IP
7. Nennen Sie ein Beispiel für einen Namensdienst.
   1. DNS-Dienst
8. Was ist beim DNS eine Top-Level-Domain? Was ist eine Domain?
   1. TLD - Länderkennzeichen und weitere (.com)
   2. Domain – Bereiche
9. Wie viele Subdomains kann eine Domain grundsätzlich besitzen? Und wie ist es in der Praxis?
   1. Beliebig viele, je nach Vertrag.
   2. Oftmals gibt es je Niederlassung einer Firma eine Subdomain.
10. Da ein einziger DNS-Server nicht die gesamte Information weltweit speichern kann, sondern nur einen Teil, ist er nicht in der Lage, alle Anforderungen zu bedienen. Was muss gemacht werden in dem Fall einer fehlenden Information auf einem Server? Wer führt diese Maßnahmen durch?
    1. Die DNS Server handeln im rekursiven Modus und geben Anfragen untereinander weiter. Der Resolver befragt mehrere DNSServer sofern der erste DNS-Server keine Antwort hat.
11. Was ist eine DNS-Cache-Only-Server?
    1. Ein DNS Cache-OnlyServer hat keine eigene Zone. Er vermerkt bzw. lernt Anfragen und Ergebnisse.
12. Was sind LDAP-Server? Wofür steht LDAP? Und was hat LDAP mit der X.500-Empfehlung zu tun?
    1. Ein Protokoll zum Verzeichniszugriff.
    2. Lightweight Directory Access Protocol
    3. Basiert auf X.500 als vereinfachte Version.
13. Inwiefern besitzen die Einträge in der LDAP-Datenbank eine Hierarchie?
    1. Eintrag - Attribut – Wert
14. Alle Einträge in der LDAP-Datenbank werden eindeutig durch den Distingushed Name (DN) identifiziert. Wie ist dieser prinzipiell aufgebaut? Skizzieren Sie dies an einem Beispiel.
    1. cn=Burkhard Messer,ou=FB4,o=HTWBerlin,c=DE
    2. common name organisational unit organisation country