

# Sisope

**1.-Kernfunktionen Betriebssysteme** - Management-Prozesse - Management der primäre Speicher-Verwaltung - **2.-Prozesse zu erklären und geben Beispiele:** \* Programm: Eine Folge von Anweisungen, einen Computer und lesen können ausführen \* Prozess: Eine Reihe von Aktivitäten und Veranstaltungen durchgeführt wird oder folgen einem bestimmten Zweck \* Parent-Prozess: Er behält die Kontrolle über alle seine Nachkommen, so dass Sie jeden Prozess zu diesem \* Kind-Prozess: die geerbte Ressourcen der Eltern zu töten, wenn sie durch die erworben werden Vater ist nicht berechtigt, Vererbung \* Prozess-und Thread: ein Prozess hat seinen eigenen Speicherbereich und läuft gleichzeitig mit anderen Prozessen, Threads innerhalb eines Prozesses teilen alle die gleichen Erinnerungen 3 .- **4.-Skript ist, dass? Und es wird verwendet,** ist ein Satz von Anweisungen, die Aufgaben ermöglichen die Automatisierung, wodurch kleine Hilfsprogramme, ist die Verwaltung verwendet für UNIX-System **beschreiben, 5.-Eltern-Kind-Prozessen** - fork: Erstellen eines Prozesses, der eine Kopie erzeugt selbst, ist diese Kopie als Kind-Prozess die Muttergesellschaft, die daraus resultierenden Prozesse sind identisch, aber mit unterschiedlichen PID - `execv` handeln: Sorgt für ein Array von Zeigern auf Zeichen Computern in 0 endet. Das erste Argument sollte und Punkt einig Befehle zum Namen der Datei, die Sie q-System ausgeführt wird: q ist eine Funktion aufrufen bzw. Funktionen **6.-alter Tisch, so dass sie der Lage sein müssen, um die Prozesse überwachen und verwalten Ressourcen-** Prozesse - Speicher-table-File Table Tisch els **Block erklären 7. bis control-gespeicherten Informationen für den Prozess** ist ein spezieller Gruppen registriert, so dass alle Informationen die Sie benötigen ungefähr wissen, um einen bestimmten Prozess-ID zählen der Prozess- Berechtigungen zugewiesen Programm-Prozess-Zustand - die Werte der CPU-Register - Eigentümer der Daten, den Stack-Speicher-Nummer Prozess-ID PID und PPID **8.-Detail erklären, wie die Durchführung der Boot-Prozess** besteht aus ständig in einem Programm den Speicher geladen in den ROM-Teil an unabhängige verblassen in den Boot-ROM "Starter-ROM, so dass dieses. **9.-nehme einen Algorithmus bevorzugt Prozesse, dass der Konsum sind die am wenigsten Zeit Prozessor in der jüngsten Vergangenheit)** Dieser Algorithmus für CPU-gebundene Prozesse oder Verfahren, begrenzt durch e / s, wenn der Prozess verbraucht eine geringe Menge an CPU-Zeit Prozesse sind dann Prozesse begrenzt durch E / S gehören zu b), was offensichtliche Nachteil dieses Algorithmus werden **Terminplanung** wird auf unbestimmte Zeit verschoben anhängigen Prozesse durch ständig begrenzten CPU **10.-erklären die Unterschiede im Grad der FCFS Algorithmen** nicht förderlich RR .- .- - nach dem Quanten-TE und TR minimiert Prozess kürzer, je nach Länge des Quanten-Queues mit Multilevel Feedback .- Wenn ein Prozess verbringt zu viel CPU-Zeit kam es Warteschlangen, um eine niedrigere Priorität Warteschlange so kurze Prozesse werden immer in hoher **Priorität, welche Funktionen können 11 .- Wert verwendet werden, mehr zur Bewertung der Effizienz der Scheduling-Algorithmen** CPU-Auslastung 1.-2.-min Timeout prom 3.-Rendite Zeit, die kann so min **12.-beschreibt die Bedienung der Planung und Nachteile** FCFS .- Verkauf: einfach zu implementieren des: nicht garantiert min prom Zeit während einige andere Prozesse sind Dateien .- vent SJF Hosios: die beste Leistung laufenden Prozesse in der Warteschlange q sind Listen, die kürzer sind, platzen des: weiß nicht genau, wie lange blast Voraus CPU nicht enteignen .- Schlot.- Priority Prozesse sind nach dem ersten des: exsiten durchgeführt werden, wenn mehrere Prozesse der Reihenfolge des Eingangs als enteignen .- sind, wenn es Prozesse mit niedriger Priorität und so ständig neue Prozesse mit Prioritäten jenseits der ersten werden können, sind Warten auf unbestimmte Zeit für q tritt nicht auf Technik wird angewendet, wenn die Alterung der Zeit, um die Prozess-Priorität Zeit ist in der Regel erhöhte RR .- Dieser Algorithmus ist in den Systemen der geteilten t q verwendet Pässe Jeder Benutzer hat den Eindruck, dass der Prozess bereits q ausgeführt und in diesem Fall ist es ratsam, q der Quantum an Zeit ist klein genug, Planung Mit mehreren Warteschlangen: oasar Prozess ermöglicht eine Warteschlange zu einem anderen, auf die Idee, für Prozesse mit unterschiedlichen Eigenschaften in ihren Ausbrüchen von CPU warten, wenn

ein Prozess CPU verbringt zu viel Zeit verbraucht wurde **13.** mit einer niedrigeren Priorität **Queue-Parallelität beschreibt die Probleme, die Prozess-Prozess auftreten können während der Ausführung** gegenseitiger Ausschluss .- zu anderen ermöglicht den Zugriff Deadlock .- kein Prozess ausgeführt wird, wartet auf einen anderen Verhungern .- Aktion, wenn ein Prozess mit niedriger Priorität bleiben, dürfen bis auf Halten **14.-beschreiben, welche Funktionen sind minimal notwendig, ein Thread zu identifizieren**, zu schaffen Thread zu identifizieren, ein Thread Thread einen Thread zu erstellen: pthread\_create sicher, dass die wichtigsten Prozess wartet dieser Thread beendet ist