1. Co to jest system operacyjny?
System operacyjny to program zarządzający pracą komputera. Program pośredniczący między komputerem a użytkownikiem. Zadania użytkownika wykonywane są jako procesy obsługiwane przez system operacyjny. System przydziela procesom zasoby komputera: dostęp do jednostki centralnej, pamięci, dyski i inne.

3. Jakie funkcje spełnia WSO i jakie podsystemy wchodzą w jego skład?
- zarządz. procesami (tworzenie,usuwanie,zawieszanie,odwiesz,IPC,synchr)
- zarządz. pamięcią (pam.główna, swap, pam. wirtualna) ; -praca sieciowa.
- zarządzanie plikami, zarządzanie przestrzenią dyskową, operacjami we/wy
- podsystem ochrony- mechanizmy kontroli dostępu procesów do zasobów OS

4. Co jest przedmiotem standaryzacji w wielodostępnych systemach operacyjnych?
Celem standaryzacji jest budowa systemów „otwartych" charakteryzujących się przenośnością aplikacji, możliwością współpracy oprogramowania na różnych maszynach, skalowalność skalowalność tj. możliwość rozbudowy sprzętowej i aplikacji bez potrzeby zmian systemu. Istota: określenie interffejsu a nie implementacji.

5. Co to jest proces?
Proces to wykonywany program, tzn. program, który został rozpoczęty i jeszcze nie zakończył się. Proces składa się z kodu programu i przydzielonemu mu obszarowi pamięci. Procesy można podzielić na procesy: systemu op., użytkownika. Wyk. w trybie użytkownika, jądra (system calls). Działają współbieżnie. Wyk. w sposób sekwencyjny. Obejmuje: zakodowany program (sekcja tekstu), bieżącą czynność (PC), stan procesu, sekcja danych.

6. Co to jest blok kontrolny procesu i do czego służy?
Jest unikalny dla każdego procesu i niezbędny do kontynuowania procesu przerwanego wcześniej. Zawiera: stan procesu, licznik rozkazów, rejestry CPU, inf. dot. planowania przydziału czasu CPU, inf. dot. zarządzania pamięcią, inf. dot. rozliczeń (zużyty czas CPU, czas rzeczywisty, ograniczenia czasu, numery konta, zadań, procesów...), inf. o stanie op. we/wy (lista otwartych plików, wykaz przydzielonych urządzeń)

7. Co oznacza współbieżne wykonywanie procesów?
W systemie wielodostępnym, pracującym z podziałem czasu jest jednocześnie wiele procesów. Procesy działają współbieżnie, co oznacza konieczność podziału obliczeniowej procesora (przełączanie) między poszczególne procesy, ponieważ każdej chwili czasu wykonywany jest tylko jeden proces.

8. Jak powstaje nowy proces?
Nowy proces jest tworzony przez wywołanie f. fork. Proces wywołujący tę fcję zwany jest procesem macierzystym, a proces utworzony-potomkiem. Potomek uzyskuje pozycję w tablicy procesów oraz PID. Tworzona jest kopia kontekstu procesu macierz., Uaktualnione zostają liczniki odwołań do plików i węzłów, z którymi związany jest potomek. PID potomka przekazywany jest procesowi macierzystemu, a wartość 0 potomnemu. Potomek dziedziczy prawa dostępu do otwartych plików, a także dzieli dostęp do plików. Procesy mają identyczne kopie segmentów instrukcji, danych i stosu. --> exec (pyt.10)

9. Jak proces jest kończony?
Proces kończy się w naturalny sposób, gdy wykonano ostatnią instr.Proces kończąc się, wywołuje fcję exit. Proces przechodzi do stanu zombie, zwalnia swoje zasoby i oczyszcza kontekst, za wyjątkiem pozycji w tablicy procesów. Proces macierzysty czekający na zak.potomka wykonuję fcję wait. Jądro szuka potomków takiego procesu, będących w stanie zombie. Jeśli znajdzie, to przekazuje PID potomka oraz parametr fcji exit i dostarcza do procesu macierzystego. Proces macierzysty uzyskuje info., który z wielu możliwych potomków zakończył pracę. Jądro zwalnia pozycję procesu potomnego w tablicy procesów.

10. Jak działa funkcja systemowa exec?
Po wywołaniu fcji fork, wywoływana jest zwykle fcja exec. Służy ona do wykonywania określonego, innego programu. Umieszcza w obszarze pamięci procesu kopię pliku wykonywalnego i rozpoczyna jego wykonywanie. Zawartość kontekstu procesu wywołującego exec zostaje zamazana. RYS: (macierzysty: środowisko1,kod prog1,zmienne lokalne1- >fork->środ1,kod1,zm.lok1, ->exec-> środ1, kod2,zm.lok2 u potomka)

11. Jak działa funkcja systemowa fork?
Funkcja fork (forking a process) tworzy nowy proces (proces potomny), który ma wykonać polecenie. Jest to kopia bieżącego procesu. (patrz pyt 8 – jak powstaje nowy proces)

12. Co to są funkcje systemowe?
Zbiór procedury w systemie, pełniących określone funkcje, o ustandaryzowanym interfejsie wy/wy (parametry-wartość zwracana przez fcję), odciążające programistę od pewnych typowych działań niskopoziomowych w systemie. Np nice(val), fstat() (sluzy do pobierania statusu pliku przechowywanego w jego i-wezle), link(o,n) (dodaje dowiazanie do pliku) Funkcje systemowe tworzą interfejs między programem a systemem operacyjnym. Można podzielić je na kilka kategorii : nadzorowanie procesów, operacje na plikach, operacje na urządzeniach, komunikacja...

13. W jakich stanach może być proces?
nowy; wykonywany : w trybie użytkownika lub jądra; gotowy do wykonywania (czeka na przydział CPU), czekający (uśpiony) – czeka na wystąpienie zdarzenia niezbędnego do dalszego wykonywania (np zakończegnie op. we/wy, np. op. dyskowej); zakończony.

14. Co to jest "shell"?
Shell jest to program, który pośredniczy między jądrem, systemem plików i programami usługowymi. Shell to interpretator poleceń.

15. Podać przykłady programów shell i ich właściwości?
Przykładowe shelle to shell Korna, shell Bourne' a, shell C, Posix.
Podstawowe funkcje: Przekazywanie sterowania do programu wybranego polecenia użytkownika, Wykonywanie wbudowanych poleceń, Dostosowywanie języka do pisania skryptów, Ustawianie środowiska pracy, Przywoływanie i edycja uprzednio wydawanych poleceń, Przeadresowanie wejścia – wyjścia poleceń, Generowanie nazw plików, Umożliwienie przetwarzania w drugim planie, umozliwienie łączenia poleceń w potok.

16. W jaki sposób program shell interpretuje polecenie?
Interpretacja programu przez shell odbywa się w nastepującej kolejności: wczytanie polecenie do wewnętrznego bufora, podzielenie polecenia na części zwane słowami –pierwsze słowo jako nazwa polecenia, shell określa znacznie każdego słowa, szukanie i przetwarzanie znaków specjalnych, zlecenie wykonania polecenia, „uśpiony" shell czeka na zakończenie polecenia, po zakończeniu wykonywania polecenia zgłasza gotowość przyjęcia nowego polecenia.

17. Na czym polega wykonywanie polecenia w tle (w drugim planie)?
Shell zleca wykonanie polecenia i wyprowadza zgłoszenie gotowości przyjęcia następnego. Nie wykonuje fcji wait.

18. Co to jest planowanie (szeregowanie) procesów?
Planowanie polega na określeniu w jakiej kolejności procesy uzyskują dostęp do zasobów komputera: procesora, pamięci operacyjnej. Procesy starając się o dostęp do procesora czy konkretnego urządzenia, czakają w kolejkach : zadań, procesów gotowych, kolejkach do urz. we/wy.

19. Podać przykłady algorytmów szeregowania procesów i wyjaśnić ich działanie?
FCFS, SJF, rotacyjne, priorytetowe.

20. Jak rozwiązuje się zagadnienie planowania procesów w systemach typu UNIX?
Kryteria planowania: wykorzystanie procesora w % czasu, przepustowość – liczba procesów kończonych w jednostce czasu, czas cyklu przetwarzania – średni czas przetwarzania procesu od chwili utworzenia do zakończenia, czas oczekiwania – średni czas oczekiwania w kolejkach, czas odpowiedzi – w systemach interakcyjnych czas między zgłoszeniem zamówienia przez użytkownika a pierwszą odpowiedzią

21. Kiedy, po co i jak wykonywany jest proces ładowania systemu?
Celem procesu ładowania jest umieszczenie systemu operacyjnego w pamięci operacyjnej i rozpoczęcie jego wykonywania. Składa się z kilku etapów: inicjalizacja i testowanie sprzętu, wczytanie i umieszczenie w pamięci bloku systemowego (blok 0 dysku), program zawarty w bloku systemowym ładuje jądro systemu operacyjnego operacyjnego z pliku systemowego (/stand/vmunix) do pamięci. Przekazuje sterowanie do pierwszej instrukcji jądra. Program jądra systemu zaczyna się wykonywać, wykrywanie i konfiguracja urządzeń, odnalezienie głównego katalogu plików, przygotowanie środowiska procesu 0. Wykonanie programu systemu jako procesu 0 wykonywanego w trybie jądra. Utworzenie procesów jądra, np. procesów zarządzania pamięcią, rozwidlenie procesu 0 (fork). Utworzony proces 1 tworzy kontekst poziomu użytkownika i do niego przechodzi, proces 1 wywołuje funkcje systemową exec wykonując program /sbin/init, proces init wczytuje plik /etc/inittab i rozmnaża procesy, inicjacja wewnętrzna struktury danych jądra, tworzenie np. listy wolnych buforów, i – węzłów, kolejek; inicjacja struktury segmentów i tablic stron pamięci, sprawdzanie głównego głównego pozostałych systemów plików (fsck), wywoływane są procesy getty monitorujące konsolę i terminale systemu komputerowego, zgodnie z deklaracjami w pliku inittab, a proces init wywołuje funkcję systemową wait monitorując zakończenie procesów potomnych. Proces init tworzy również procesy demony.

22. Jaki proces ma PID = 1, jakie zadania wykonuje?
Przodkiem wszystkich procesów procesów systemie jest proces o nazwie init oraz identyfikatorze PID równym 1, tworzony w czasie startu systemu. Proces init odpowiada za powoływanie wielu innych procesów (zawsze na drodze fork-and-exec), wśród których wiele wykonuje program getty. Każdy z procesów getty związane jest z innym terminalem i odpowiada za wyświetlenie na danym terminalu znaku zachęty do logowania oraz rozpoczęcia sesji.

23. Co to są pliki i jakie typy plików występują w systemie UNIX?
Pliki to jednostki logiczne przechowywanej informacji, niezależne od właściwości fizycznych urządzeń pamięciowych. Zwykle w plikach przechowywane są programy lub dane.
W Unixie, inaczej niż w innych systemach operacyjnych, pojęcie pliku ma kluczowe i centralne znaczenie. Na przykład polecenia są wykonywalnymi plikami, przechowywanymi w z góry przewidywalnych miejscach drzewa katalogowego. Systemowe przywileje i prawa dostępu kontrolowane są prawie wyłącznie przez prawa dostępu do odpowiednich plików.
Nawet realizacja operacji we/wy do urządzeń i do plików z punktu widzenia użytkownika jest nierozróżnialna. Także komunikacja między procesami odbywa się z użyciem struktur reprezentowanych w systemie jako pliki.
Podstawowe typy plików: pliki zwykłe, pliki specjalne (pliki urządzeń), katalogi, dowiązania symboliczne, potoki nazwane FIFO (ang. named pipe) (do IPC), gniazda (ang. UNIX – domain sockets) (do IPC)

24. Co to jest i-węzeł?
Jako i – węzeł rozumieć będziemy przechowywaną na dysku strukturę, opisującą atrybuty pliku, włączając w to fizyczną lokalizację bloków danych. W momencie przygotowywania danej partycji dyskowej na potrzeby przechowywania plików, od razu tworzona jest pewna określona ilość i – węzłów. Stanowi ona limit wyznaczający maksymalną liczbę wszystkiego rodzaju plików, w tym plików zwyczajnych, katalogów, plików specjalnych, łączników, które będą mogły być utworzone na tym fragmencie dysku. Standardowo na jeden plik przypada od 2 do 8 kB bloków, stąd liczba i – węzłów wyliczona jest automatycznie na podstawie rozmiaru danego fragmentu dysku i w zasadzie jest zupełnie wystarczająca. Wszystkie i–węzły mają unikatowe numery. Z każdym plikiem związany jest opisujący go i–węzeł, rezerwowany w momencie tworzenia danego pliku.

25. Jakie informacje są przechowywane w i-węźle?
W i–węźle przechowywane są następujące informacje o plikach: typ pliku (plik zwykły, katalog itd. lub 0, gdy i – węzeł nie jest używany, prawa dostępu do plików, liczba dowiązań do pliku, UID/GID, rozmiar w B, czas ostatniego dostępu, ost. modyfikacji pliku, czas ostatniej modyfikacji i – węzła, 12 wskaźników adres. bezpośr., wskaźnik zawierający adres bloku, w którym przechowywane są adresy bloków z adresami bloków z danymi (2st.), skaźnik wykorzystywany w adresowaniu pośrednim 3st

26. Jakie informacje przechowywane są w pliku typu "katalog"?
Katalogi służą do powiązania nazw plików z danymi znajdującymi się na dysku. W każdym katalogu może znajdować się pewna liczba plików i innych katalogów (podkatalogów). Katalog jest przechowywany jak plik zwykły i (w uproszczeniu) ma postać tabeli o dwóch kolumnach. Każdy wiersz tej tabeli zawiera nazwę znajdującego się w katalogu pliku (lub podkatalogów) oraz numer i-węzła pliku. 4B nr i-węzła, 2B dług.elementu katalogu, 2B długość nazwy pliku, 14-255B nazwa pliku.

31. Czym różni się przydział ciągły miejsca na dysku od przydziału listowego?
Przydział ciągły: Plik zajmuje ciąg kolejnych bloków. Adres początkowy+rozmiar. Zalety: dostęp do kolejnych bloków nie wymaga ruchu głowic, b. łatwy dostęp do plików. Wady: zewn. fragmentacja. Zarówno dostęp bezpośr. jak i sekwencyjny.
Przydział listowy: Plik stanowi listę powiązanych bloków które mogą znajdować się w dowolnym miejscu na dysku. Na końcach bloków znajdują się informacje o adresie nast. bloku z danymi.

32. Co jest tablica FAT?
Przydział listowy+dodatkowa tablica o tylu pozycjach, ile jest bloków na dysku. Na dysku wydzielona jest sekcja zawierająca tablicę. W tablicy jest po 1 pozycji dla każdego bloku. Tablica jest indeksowana za pomocą nr bloków i zawiera listę powiązań. Czyli pozycja w tab. indeksowana przez nr odpowiedniego bloku zawiera numer bloku następnego w danym pliku. W pozycji odpowiadającej blokkowi ostatniemu jest EOF. Wady: brak dostępu bezpośr..uszkodzenie tabl.FAT powoduje utratę plików. Zalety: rozwiązany problem zewn. fragm., łatwo można powiększać pliki.

33. Porównać przydział listowy miejsca na dysku z przydziałem indeksowym?
Listowy -> lista powiązanych bloków na końcu jednego jest adres następnego jeśli potrzeba dostęp sekwencyjny. Indeksowy -> każdy plik ma własny plik indeksowy. Blok indeksowy pliku zawiera adresy kolejnych bloków danych pliku. Dostęp bezpośredni.

35. Wyjaśnić mechanizm wymiany (swapping) procesów?
Wymiatanie (swapowanie) jest to mechanizm, za pomocą którego UNIX rozdziela dostępną pamięć między aktualnie aktywne procesy, gdy ich łączne wymagania pamięciowe przekraczają fizyczną pamięć operacyjną.
Swapowanie polega na zapisaniu na dysku całego procesu, a tym samym zwolnieniu całej zawiązanej z nim pamięci. Proces przeniesiony do obszaru swap dysku musi zostać ponownie wczytany do pamięci, by można było go kontynuować.

36. Wyjaśnić mechanizm stronicowania na żądanie?
Stronicowanie (paging) wymaga przenoszenia części pamięci przydzielonej procesowi – w jednostkach zwanych stronami (page) – na dysk, aby zwolnić pamięć potrzebną dla tego bądź innego procesu.
Stronicowanie na żądanie wykorzystuje zasadę lokalności odwołań i do pamięci przesyłane są strony niezbędne w danej chwili lub te, które niebawem mogą się okazać niezbędne. Strony mogą znajdować się na dysku, w pamięci głównej lub w obszarze wymiany. Jeśli niezbędna jest strona znajdująca się na dysku, generowany jest błąd strony i odpowiednie przerwanie. Wykonywanie procesu jest wstrzymywane, po czym odnajdywana jest wolna ramka, do której przypisywana jest potrzebna strona.

37. Wyjaśnić mechanizm segmentacji?
Segmenty to określone fragmenty programu. Do podstawowych zalet segmentacji należą: możliwość powiązania ochrony pamięci z wybranymi segmentami oraz współdzielenie wybranych segmentów przez różne procesy. Na przykład ustawienie bitu ochrony dla segmentów kodu (tylko do odczytu) lub współdzielenie kodu edytora (np. vi) przez procesy edycji wielu jednocześnie pracujących użytkowników. W tym przypadku obrazy procesów użytkowników zawierają, w uproszczeniu, segmenty danych i stosu oraz wskaźniki do właściwego miejsca w segmencie programu. Brak fragmentacji zewn.
Aby odwołać się do odpowiedniego segmentu, procesy korzystają z tablicy segmentów. Każdy element tej tablicy można przedstawić w postaci pary, adres bazowy danego segmentu oraz offset. Tablicę segmentów przechowuje się zwykle w pamięci głównej, a tylko pewną liczbę elementów tej tablicy przechowuje się w rejestrach (TLB-asocjacyjne)

38. Podać przykłady algorytmów wymiany stron?
FIFO, optymalny, LRU (Least Recently Used)

39. Kiedy występuje błąd strony i jak jest obsługiwany?
Błąd strony (page fault) pojawia się wtedy, gdy procesowi potrzebna jest strona pamięci, która nie rezyduje w fizycznej pamięci operacyjnej i musi zostać wczytana z dysku.

41. Jakie są zadania podsystemu wejście - wyjście? (ŹLE)
Zarządzanie przestrzenią nazw plików, nadzoruje przebieg dostępu do plików urządzeń, nadzoruje poprawność operacji, przydziela miejsca w systemie, urządzenia, planuje operacje wejścia wyjścia, buforowanie, przechowywanie, obsługa błędów

42. Co to są pliki specjalne i do czego służą?
Pliki specjalne w UNIXie umożliwiają operacje we/wy do urządzeń. Pliki te przechowywane są standardowo w katalogu /dev (oraz w jego podkatalogach – System V). plikach specjalnych nie przechowuje się żadnych danych. Pliki te charakteryzuje sposób działania urządzenia, wskazują miejsce podłączenia urządzeń do systemu oraz zapewniają dostęp programów obsługi urządzeń.
Blokowe/znakowe. Numer glowny i drugorzedny.

43. Co to jest tablica rozdzielcza urządzeń we-wy, jakie informacje zawiera i do czego służy?
Interfejs między systemem operacyjnym a programami obsługi urządzeń.
Tablica rozdzielcza urządzeń blokowych -> open_devX, close_devX, strategia_devX > transmisja danych między buforem a urządzeniem
Tablica rozdzielcza urządzeń znakowych -> open_rdev, close_rdev, read_rdev, write_rdev, ioctl_rdev > input output control

45. Jakie są typy plików specjalnych?
Pliki specjalne znakowe (mknod -c lub u), Pliki specjalne blokowe (b) (buforowane), FIFO (p).
Termin "plik specjalny" ("special file") posiada w Uniksie znaczenie techniczne: coś, co potrafi tworzyć lub odbierać dane. Zwykle odpowiada on jakiemuś elementowi sprzętowemu, np. drukarce czy dyskowi. (Pliki te są na ogół tworzone podczas konfiguracji systemu.) To właśnie polecenie mknod tworzy pliki tego rodzaju. Z takich urządzeń może być jednorazowo odczytywany albo pojedynczy znak albo "blok" (wiele znaków), zatem mówimy, że istnieją "blokowe pliki specjalne" i "znakowe pliki specjalne".

46. Jakie informacje są zapisane w plikach specjalnych i do czego służą?
Pliki specjalne znakowe odpowiadają za realizację znakową, czyli surowego dostępu do urządzeń. Pliki specjalne blokowe umożliwiają dostęp do bloków. Pliki specjalne znakowe używane są w celu wykonania niebuforowanego przesyłania danych na i z urządzenia, podczas gdy pliki specjalne blokowe wykorzystywane są wtedy, gdy dane mają być zapisywane bądź odczytywane porcjami, określanymi jako bloki.

47. Jaką rolę pełnią podprogramy obsługi urządzeń? (drivery)
Podprogram obsługi urządzenia jest to taki zbiór funkcji z jądra systemu, do którego dostęp otrzymuje się poprzez tablice rozdzielcze urządzeń.
Podprogramy obsługi urządzeń mają jednak pewną cechę, która w dość naturalny sposób wyróżnia je spośród innych części jądra systemu. Cecha tą jest metoda dostępu do ich funkcji spoza jądra. Normalnie UNIX udostępnia użytkownikom swoje usługi poprzez mechanizm funkcji systemowych. Tymczasem dostęp do podprogramów obsługi urządzeń otrzymuje się przez system plików.

Termin "podprogram obsługi urządzenia" pochodzi stąd, że często funkcje przez ten podprogram udostępniane są to funkcje pozwalające wykonywać operacje wejścia-wyjścia na fizycznych urządzeniach, takich jak dysk twardy lub drukarka.