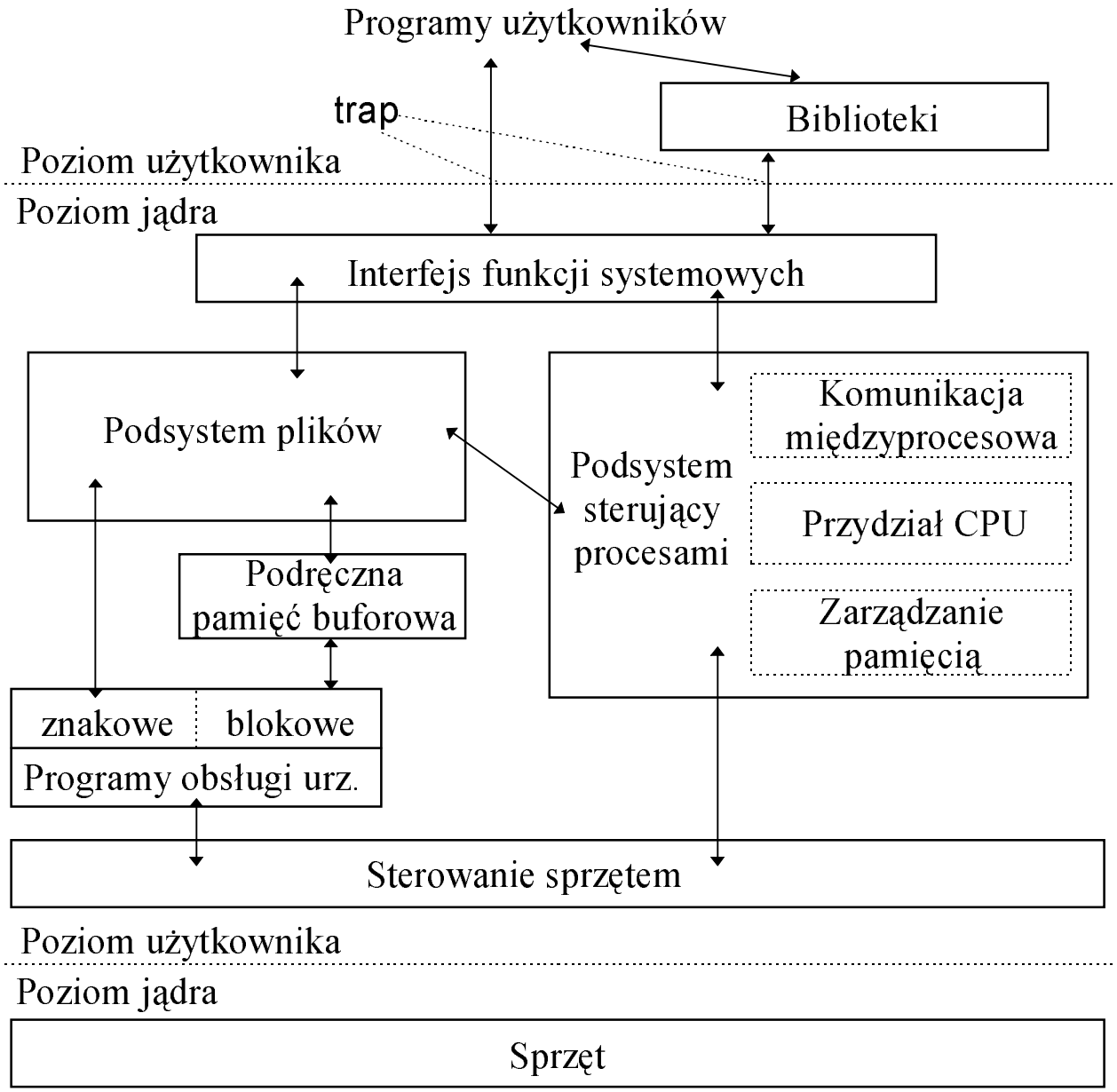
System operacyjny Unix

Budowa jądra

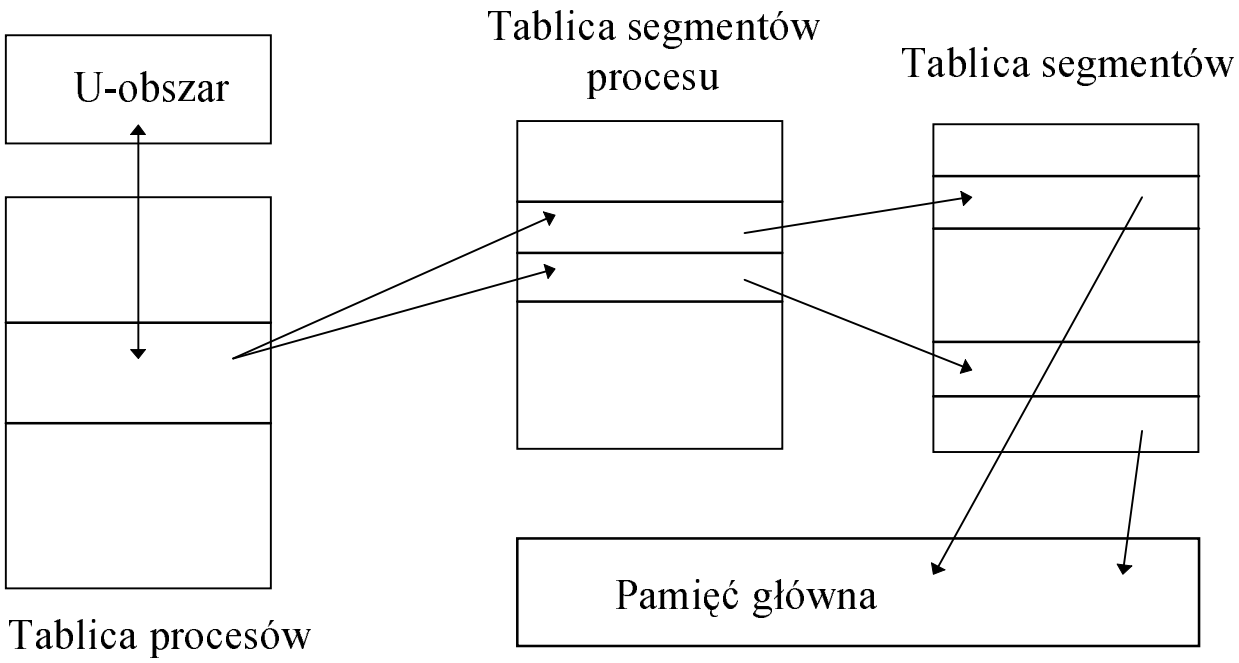


Procesy

- Każdy proces ma unikatowy identyfikator (*pid*, ang. *process identifier*)
- Wyróżnione procesy:
  proces 0 - specjalny proces tworzony podczas inicjalnego ładowania systemu; po utworzeniu *procesu 1* staje się procesem wymiany (ang. *swapper*)

proces 1 - (ang. *init*) przodek wszystkich pozostałych procesów

Struktury danych związane z procesem



- Pozycja w tablicy procesów i u-obszar zawierają informacje sterujące oraz opis stanu procesu
- Pozycja w tablicy procesów:
  - pole stanu
  - identyfikator użytkownika będącego właścicielem procesu (uid, ang. *user identifier*)
  - gdy proces jest w stanie uśpionym, to deskryptor zdarzenia
- U-obszar zawiera informacje niezbędne podczas wykonywania procesu:
  - wskaźnik do pozycji w tablicy procesów
  - argumenty bieżącego wywołania f-cji systemowej, wartości przekazywane z f-cji i kody błędów
  - deskryptory otwartych plików
  - bieżący katalog i bieżący korzeń
  - ograniczenia na rozmiar procesu i pliku

- Jądro ma bezpośredni dostęp jedynie do u-obszaru wykonywanego procesu (lecz nie do u-obszarów innych procesów)
- Pozycja w tablicy segmentów opisuje atrybuty segmentu:
  - instrukcje czy dane
  - dzielony czy prywatny
  - położenie w pamięci operacyjnej
- Dodatkowy poziom pośredniości (tablica segmentów) umożliwia współdzielenie segmentów (exec, fork)
- Kontekst procesu to jego stan zdefiniowany poprzez:
  - instrukcje procesu
  - wartości zmiennych globalnych i struktur danych
  - wartości rejestrów maszynowych
  - zawartość pozycji w tablicy procesów i u-obszarze
  - zawartość stosu użytkownika i stosu systemowego
- System wykonuje się w kontekście wykonywanego procesu
- Zmiana kontekstu (rozpoczęcie wykonywania innego procesu) a zmiana trybu pracy procesora (z systemowego na użytkowy lub odwrotnie)
- Stany procesu

Szeregowanie procesów

- Strategia karuzelowa, wielopoziomowa, ze sprzężeniem zwrotnym, kolejki priorytetowe
- Do wykonania wybiera się proces o najwyższym priorytecie spośród procesów gotowych do wykonania i załadowanych do pamięci
- Dwie klasy priorytetów: systemowe i użytkowe
  - procesy z piorytetami użytkowymi są wywłaszczane tuż przed powrotem z trybu jądra do trybu użytkowego
  - procesy z priorytetami systemowymi otrzymują piorytety w chwili zasypiania

- Jądro wylicza priorytet procesu w określonych stanach:
  - Przydziela priorytet procesowi, który ma przejść do stanu uśpienia. Stała wartość priorytetu odpowiada przyczynie uśpienia (dobór wartości - tak by zminimalizować liczbę konfliktów)
  - Jądro uaktualnia priorytet procesu, który wraca z trybu jądra do trybu użytkownika. Proces mógł wcześniej przejść do stanu uśpienia zmieniając swój priorytet na systemowy - teraz trzeba go obniżyć do użytkowego. Ponadto zmniejsza mu priorytet w stosunku do innych procesów, gdyż proces właśnie korzystał z zasobów systemu
  - Procedura obsługi przerwania zegarowego uaktualnia priorytety wszystkich procesów w trybie użytkowym w odstępach 1 sekundowych i powoduje, że jądro wykonuje algorytm szeregowania, by zapobiec monopolizowaniu CPU przez jeden proces

System plików w Unixie

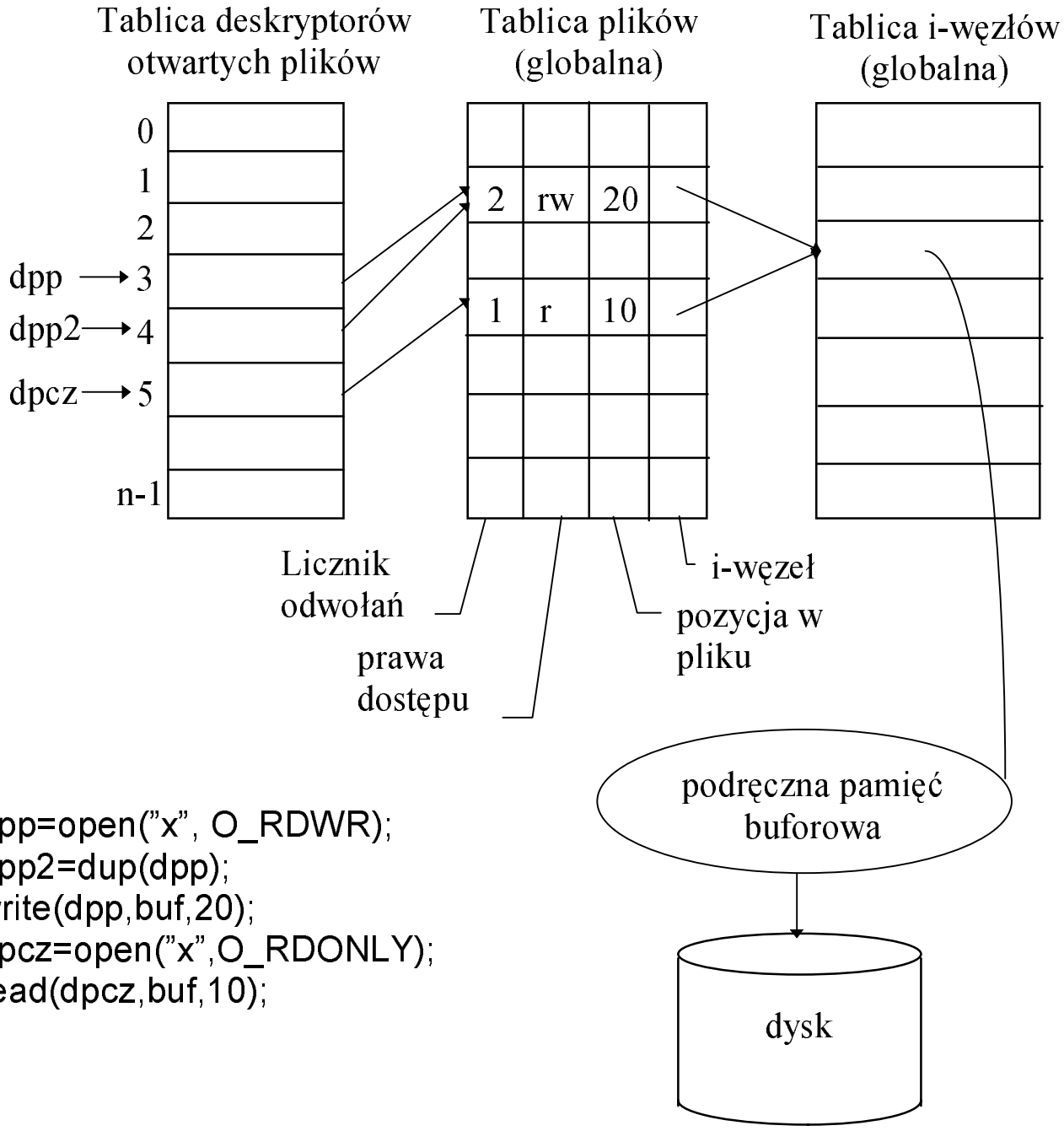
- Plik jest tablicą bajtów
- Plik można rozszerzyć pisząc za jego końcem
- Z punktu widzenia użytkownika f-cje systemowe do obsługi plików są synchroniczne
- Każdy plik ma swoją nazwę ścieżkową (jedną lub więcej → ***dowiązanie***, ang. *link*)
- Każdy plik ma unikatowy ***i-numer*** i ***i-węzeł***

Katalogi

- Nazwa pliku i i-numer są zapamiętane w ***katalogu***
- Katalogi w systemie plików mają strukturę grafu

unix	117
etc	4
home	18
pro	36
dev	93

Systemowe struktury danych do obsługi plików




```
dpp=open("x", O_RDWR);
dpp2=dup(dpp);
write(dpp,buf,20);
dpcz=open("x",O_RDONLY);
read(dpcz,buf,10);
```

I-węzły

- I-węzły są zapisane na dysku i wczytywane przez jądro do pamięci
- I-węzeł na dysku zawiera:
  - uid właściciela pliku
  - typ pliku (zwykły, katalog, specjalny znakowy lub blokowy, łącze nazwane)
  - prawa dostępu do pliku
  - czas: ostatniej modyfikacji, ostatniego dostępu, ostatniej modyfikacji i-węzła
  - liczba dowiązań do pliku
  - adresy bloków dyskowych pliku (pliki mogą być „dziurawe")
  - rozmiar pliku
- Kopia i-węzła w pamięci zawiera ponadto:
  - status i-węzła w pamięci: czy zablokowany, czy są procesy czekające na zdjęcie blokady, czy kopia w pamięci była modyfikowana, czy plik jest punktem zamontowania
  - logiczny numer urządzenia systemu plików zawierającego plik
  - numer i-węzła (indeks w tablicy i-węzłów na dysku)
  - wskaźniki do innych i-węzłów w pamięci (kolejek mieszających, listy wolnych). Wybór kolejki mieszającej - wg logicznego numeru urządzenia i numeru i-węzła. Jądro zawiera co najwyżej jedną kopię i-węzła dyskowego w pamięci, ale i-węzły mogą być równocześnie w kolejce mieszającej i na liście wolnych
  - licznik odwołań wskazujący liczbę aktywnych instancji pliku (ile razy plik został otwarty)

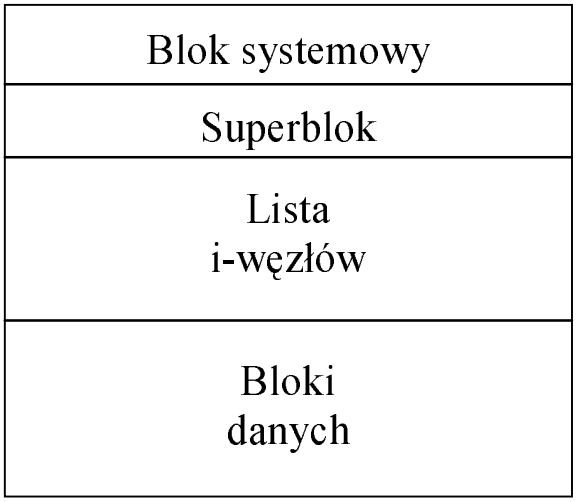
- I-węzeł jest na liście wolnych tylko wtedy, gdy licznik odwołań = 0 (tzn. można przydzielić to miejsce w pamięci na inny i-węzeł dyskowy)

- Dostęp do i-węzłów (algorytm iget)
  Jądro szuka i-węzła w odpowiedniej kolejce mieszającej. Jeśli nie znajduje, to przydziela i-węzeł z listy wolnych i zakłada na niego blokadę. Czyta i-węzeł z dysku (alg. *bread*). Zna nr i-węzła i wie ile i-węzłów mieści się w jednym logicznym bloku pliku. Usuwa i-węzeł z listy wolnych, umieszcza go w kolejce mieszającej i ustawia licznik odwołań na 1. Kopiuje info z i-węzła dyskowego do i-węzła w pamięci i przekazuje i-węzeł z założoną blokadą.
  Blokada jest założona w czasie wykonania f-cji systemowej, by uniemożliwić innym procesom równoczesne korzystanie z i-węzła. Zdejmuje się ją po zakończeniu wykonania f-cji systemowej, dzięki czemu jest możliwe współdzielenie pliku.
  Jeśli lista wolnych i-węzłów jest pusta, to jądro zgłasza błąd (czekanie na zwolnienie jakiegoś i-węzła byłoby zbyt ryzykowne).
  Jeśli i-węzeł jest w pamięci, ale zablokowany, to proces zasypia w oczekiwaniu na zdjęcie blokady

- Zwalnianie i-węzłów (algorytm iput)
  Jądro zmniejsza licznik odwołań. Jeśli otrzymuje 0, to zapisuje i-węzeł na dysk (o ile był modyf.). Umieszcza i-węzeł na liście wolnych. Jeśli liczba dowiązań do pliku spadła do 0, to zwalnia bloki dyskowe pliku oraz i-węzeł

- Konwersja nazwy ścieżkowej na i-węzeł (alg. namei)

Organizacja systemu plików na dysku



- **Superblok** (blok informacyjny, identyfikacyjny)
  Zawartość:
  - rozmiar systemu plików
  - liczba wolnych bloków
  - lista wolnych bloków
  - indeks nast. wolnego bloku na liście wolnych bloków
  - rozmiar listy i-węzłów
  - liczba wolnych i-węzłów
  - lista wolnych i-węzłów
  - indeks nast. wolnego i-węzła na liście wolnych i-węzłów
- Przydział i-węzła dla nowego pliku (algorytm ialloc)
  System plików utrzymuje liniową listę i-węzłów. I-węzeł jest wolny, jeśli pole typu jest wyzerowane. Jądro sprawdza czy inne procesy nie zablokowały dostępu do listy wolnych i-węzłów w superbloku.
  Jeśli lista numerów i-węzłów nie jest pusta, to jądro przydziela nast. numer, wpp przeszukuje dysk i umieszcza w superbloku tyle numerów wolnych i-węzłów, ile się zmieści. Zapamiętuje najwyższy numer znalezionego i-węzła. Nast. razem przeszukiwanie rozpocznie się od tego miejsca.

- Zwalnianie i-węzła
  Jeśli lista wolnych nie jest pełna, to dołącza do niej numer, wpp porównuje ten numer z zapamiętanym numerem, zachowuje mniejszy z nich, a pozostały zapomina.

Podręczna pamięć buforowa

- Jądro próbuje minimalizować częstość operacji dyskowych przez utrzymywanie puli wewnętrznych buforów zawierających ostatnio używane bloki dyskowe
- **Bufor** składa się z *nagłówka* i *danych*. Blok dyskowy może się równocześnie znajdować tylko w jednym buforze. Nagłówek bufora zawiera:
  - logiczny numer urządzenia (systemu plików)
  - numer bloku
  - wskaźnik do tablicy danych
  - pole stanu: bufor jest zablokowany, bufor zawiera aktualne dane, trzeba zapisać zawartość bufora na dysk przed przydzieleniem bufora dla innego bloku (*opóźniony zapis*), jądro właśnie czyta/pisze zawartość bufora, proces czeka na zwolnienie bufora
  - wskaźniki
- Lista wolnych buforów jest uporządkowana w kolejności LRU. Jest to lista z podwójnymi dowiązaniami. W momencie inicjacji systemu zawiera wszystkie bufory. Wolny bufor jest pobierany z początku listy, a dołączany na koniec (sporadycznie na początek). Bufory są pamiętane w kolejkach mieszających (wg numeru urządzenia i bloku). Każda lista jest cykliczna i ma podwójne dowiązania
- Pozycja bufora na liście jest nieistotna. Bufor może się znaleźć tylko w jednej takiej kolejce, ale ponadto na liście wolnych (tylko wtedy, gdy nie jest zablokowany)

- Przydział bufora na blok dyskowy:

1. Jądro znajduje blok w odpowiedniej kolejce mieszającej i bufor jest wolny.
   Zaznacza bufor jako zajęty i usuwa go z listy wolnych
2. Jądro nie znajduje bloku w kolejce, więc przydziela bufor z listy wolnych.
   Usuwa bufor z listy wolnych i umieszcza we właściwej kolejce mieszającej
3. Jądro nie znajduje bloku w kolejce, a przy próbie przydzielenia bufora z listy wolnych znajduje bufor z oznaczeniem do opóźnionego zapisu. Inicjuje asynchroniczny zapis bloku na dysk i przydziela inny bufor. Po zakończeniu zapisu bloku umieszcza bufor na początku listy wolnych
4. Jądro nie znajduje bloku w kolejce, a lista wolnych buforów jest pusta.
   Proces zasypia w oczekiwaniu na zwolnienie bufora. Po obudzeniu ponownie szuka wolnego bufora
5. Jądro znajduje blok w kolejce, ale bufor jest właśnie zajęty.
   Proces zasypia w oczekiwaniu na zdjęcie blokady, po obudzeniu ponownie sprawdza, czy bufor jest dostępny oraz czy zawiera właściwy blok

- Bufor ma założoną blokadę na czas wykonania f-cji systemowej. Procesy w trybie użytkownika nie kontrolują bezpośrednio przydziału buforów, nie mogą więc celowo ich przetrzymywać. Jądro przestaje kontrolować bufor tylko podczas oczekiwania na zakończenie we-wy między buforem a dyskiem

- <u>Czytanie i pisanie bloków dyskowych</u>
  Jeśli blok jest w buforze, to nie ma fizycznej operacji we-wy. Jeśli go nie ma, to jądro wywołuje procedurę obsługi

urządzenia, która inicjuje operację we-wy i zasypia w oczekiwaniu na zakończenie. Kontroler dysku generuje przerwanie budząc śpiące jądro - zawartość bloku dyskowego jest już w buforze
Czytanie z wyprzedzeniem
Opóźniony zapis (co innego niż *asynchroniczny zapis*)

- **Wady i zalety podręcznej pamięci buforowej:**
  - umożliwia jednorodny dostęp do dysku
  - dążenie do minimalizacji liczby transmisji dyskowych (im większa pula buforów, tym mniej transmisji, ale też mniej pamięci dla procesów)
  - system jest b. podatny na awarie, ze względu na opóźniony zapis
  - dodatkowe kopiowanie między pamięcią użytkownika i jądra jest wadą w przypadku dużych porcji danych, ale zaletą w przypadku małych (bo mniej operacji dyskowych)

Inicjalne ładowanie systemu

- Procedura inicjalnego ładowania systemu wygląda nieco inaczej w systemach o różnych architekturach
- Cel: umieszczenie kopii SO w pamięci komputera i rozpoczęcie wykonywania
- Operacja ta jest wykonywana w kilku etapach (stąd nazwa ang. *bootstrap*)
- Administrator może ustawić zbiór przełączników na konsoli określających adres specjalnego programu ładującego lub wcisnąć pojedynczy przycisk, który powoduje ładowanie programu inicjującego za pomocą mikrokodu. Program ten może się składać z kilku instrukcji, które powodują rozpoczęcie wykonywania innego programu (zapisanego w ROM)

- W Unixie procedura ładująca czyta blok systemowy z dysku do pamięci. Program zawarty w bloku systemowym ładuje jądro z systemu plików (np. z pliku /unix). Po załadowaniu sterowanie przekazuje się do adresu startowego jądra.

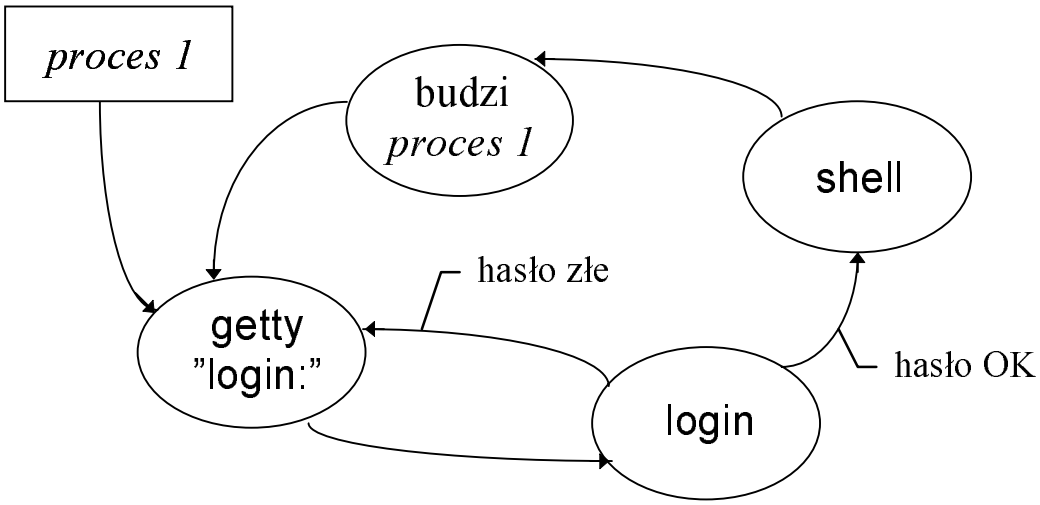
Czynności wykonywane przez jądro:

1. Inicjuje wszystkie swoje wewnętrzne struktury danych (np. listy wolnych buforów i i-węzłów, kolejki mieszające buforów i i-węzłów, tablice stron, segmentów)
2. Montuje główny system plików (w katalogu */*)
3. Przygotowuje środowisko dla *procesu 0*: tworzy u-obszar, pozycję 0 w tablicy procesów, definiuje korzeń systemu plików jako katalog bieżący, itp.
   System wykonuje się teraz jako *proces 0*
4. Tworzy proces potomny (fork) → *proces 1*.
   Proces 1, wykonując się w trybie jądra, tworzy swój kontekst poziomu użytkownika, przydzielając obszar danych i przyłączając go do swojej przestrzeni adresowej. Kopiuje do niego swój przyszły kod. Teraz „wraca" z trybu jądra do trybu użytkownika i zaczyna wykonywać swój skopiowany kod.
   Proces 1, w odróżnieniu od *procesu 0*, jest procesem poziomu użytkownika. Tekst *procesu 1* składa się z wywołania exec dla programu /etc/init.
5. *Proces 1* jest odpowiedzialny za inicjowanie nowych procesów. Czyta plik /etc/inittab zawierający info o tym, jakie procesy należy utworzyć. M.in. w trybie wieloużytkownikowym tworzy procesy getty, które monitorują linie terminalowe (getty są tworzone z opcją spawn, tzn. gdy proces umrze, to jest tworzony na nowo)

6. *Proces 1* wykonuje wait i czeka na śmierć swoich procesów potomnych (także tych odziedziczonych po innych procesach)

Wchodzenie do systemu:

1. getty jest tworzone dla każdej aktywnej linii terminalowej (*proces 1* wykonuje w tym celu fork i exec)
2. getty pisze login: i czeka na wprowadzenie identyf.
3. proces login prosi o hasło, czyta je, sprawdza. Jeśli złe, powrót do 2. Jeśli dobre, wykonuje exec dla interpretatora poleceń
4. Interpretator w chwili zakończenia budzi *proces 1*, który tworzy nowy getty



Rodzaje procesów w Unixie:

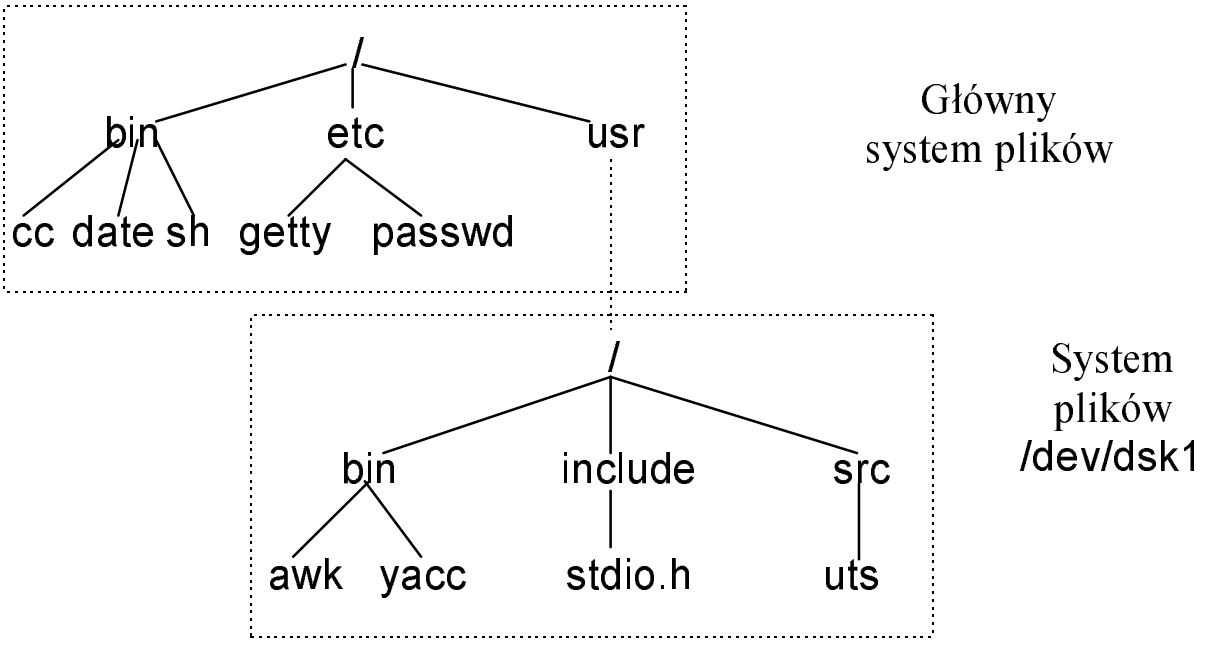
- **Procesy użytkowe**
- **Procesy-demony**. Nie są związane z żadnym użytkownikiem, wykonują ogólne funkcje systemowe, np. administrowanie i nadzór nad siecią, wykonywanie czynności uzależnionych czasowo, nadzór nad drukowaniem. Wykonują się w trybie użytkowym
- **Procesy jądra**. Wykonują się w trybie jądra. Tworzy je *proces 0*, który nast. przejmuje funkcje procesu wymiany. Są podobne do demonów, gdyż wykonują ogólne funkcje

systemowe, ale mają większe uprawnienia, bezpośredni dostęp do danych jądra itp. Mniej elastyczne, gdyż zmiany w tych procesach wymagają rekompilowania jądra

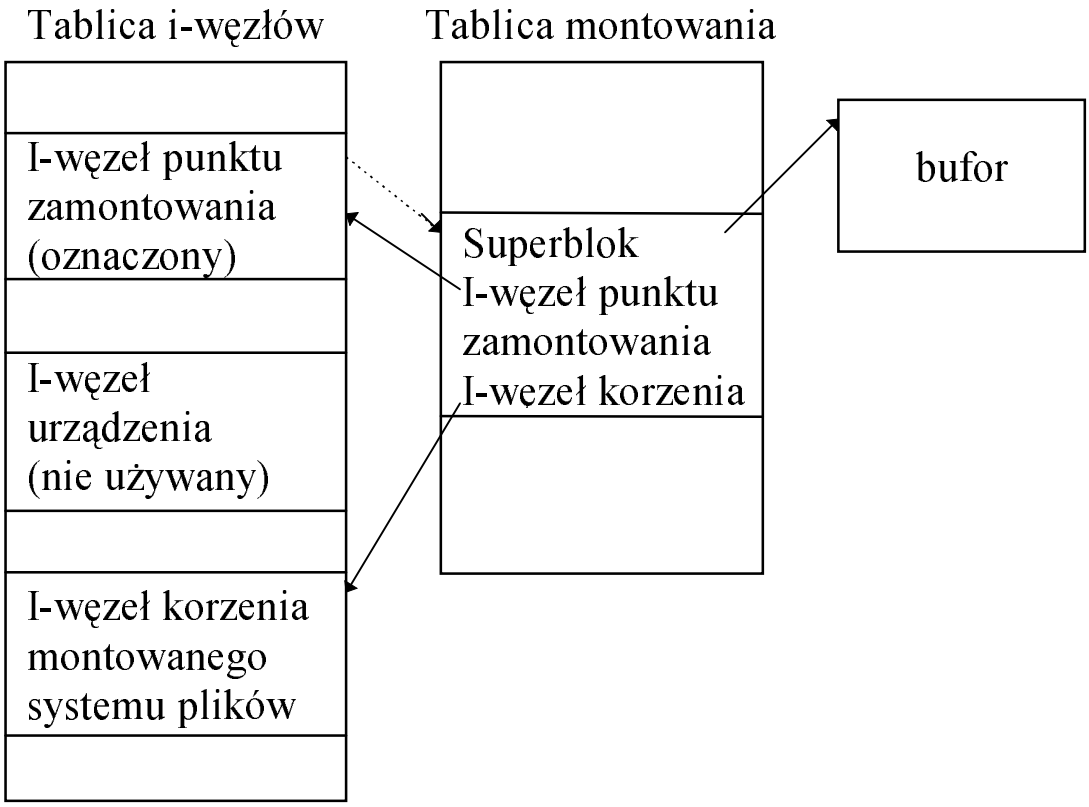
Montowanie i demontowanie systemu plików

- Fizyczna jednostka dyskowa może się składać z kilku **sekcji logicznych**. Podziału dokonuje podprogram obsługi dysku (ang. *disk driver*). Każda sekcja ma swoją nazwę pliku urządzenia (**plik specjalny**). Procesy mogą odczytywać dane z sekcji otwierając taki plik, wykonując na nim read, write, traktując go jako ciąg bloków dyskowych. Sekcja dysku może zawierać **logiczny system plików** (z blokiem systemowym, superblokiem itp.). Polecenie mount dołącza taki system plików do istniejącego systemu, a umount odłącza. Mount pozwala więc użytkownikom sięgać do danych na dysku poprzez system plików, a nie jako do ciągu bloków

mount(”dev/dsk1”, ”/usr”, 0)



- Jądro utrzymuje **tablicę montowania**. Każda pozycja zawiera:
  - numer urządzenia, który identyfikuje montowany system plików
  - wskaźnik do bufora zawierającego superblok
  - wskaźnik do i-węzła korzenia (/ dla /dev/dsk1)
  - wskaźnik do i-węzła katalogu, który jest **punktem zamontowania** (usr)



- Przechodząc wzdłuż ścieżkowej nazwy pliku jądro „gładko” przekracza punkty zamontowania
- Dla użytkownika katalog punktu zamontowania i korzeń zamontowanego systemu plików są logicznie równoważne
- Tylko nadzorca może montować i demontować systemy plików

umount(”dev/dsk1”)

- Znajduje i-węzeł urządzenia
- Znajduje odpowiadającą mu pozycję w tablicy montowania
- Upewnia się, że żaden plik z tego systemu plików nie jest w użyciu (przeglądając tablicę i-węzłów). Takie pliki mają dodatni licznik odwołań i obejmują: bieżące katalogi procesów, pliki otwarte i nie zamknięte, pliki z dzielonym kodem, który jest właśnie wykonywany. Jeśli są takie pliki, to zgłasza błąd.
- Wypisuje na dysk bloki zaznaczone do opóźnionego zapisu
- Usuwa dzielony kod, który jest nieaktywny
- Uaktualnia i-węzły
- Zwalnia i-węzeł korzenia zamontowanego systemu plików
- Woła program obsługi urządzenia w celu zamknięcia urządzenia
- Przegląda pamięć buforową unieważniając bufory zawierające bloki zdemontowanego systemu plików (przesuwa bufory na początek listy wolnych)
- Zdejmuje flagę zamontowania z punktu zamontowania i zwalnia i-węzeł
- Zwalnia pozycję w tablicy montowania

Nie można demontować głównego katalogu plików (/), gdyż zawiera on programy i pliki niezbędne do poprawnej pracy systemu