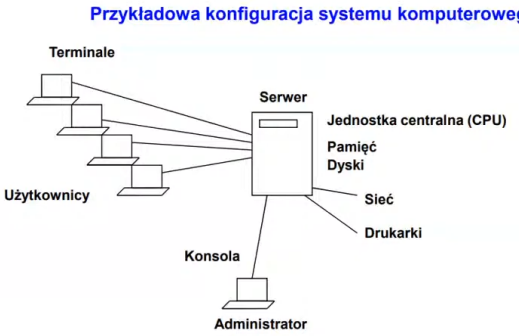
Systemy operacyjne 1

Igor Nowicki

16 lutego 2021

Spis treści

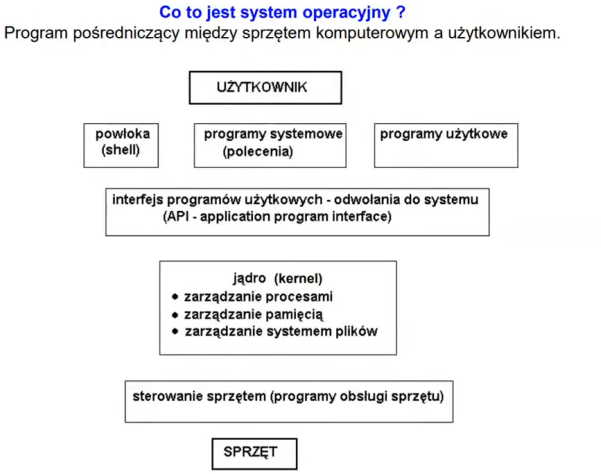
1 Wprowadzenie do wielodostępnych systemów operacyjnych



System operacyjny: program zarządzający pracą komputera

Zadania użytkownika wykonywane są jako procesy obsługiwane przez system operacyjny. System przydziela procesom zasoby komputera: dostęp do jednostki centralnej, pamięć, dyski, i inne.

Niezbędny administrator - osoba odpowiedzialna za sprawne działanie i bezpieczeństwo systemu. Również opiekun użytkowników, zakłada konta użytkowników, służy pomocą i radą.



Ani użytkownik, ani proces wywoływany bezpośrednio przez użytkownika nie ma bezpośredniego dostępu do sprzętu. Jeśli użytkownik chce skorzystać ze sprzętu, musi wywołać odwołanie poprzez API.

1.1 Podstawowe cechy systemu operacyjnego

1.2 Części składowe systemu operacyjnego.

1.3 Struktura systemu operacyjnego

1.4 Usługi systemu operacyjnego.

1.5 Co to jest system operacyjny?

Program pośredniczący między sprzętem komputerowym a użytkownikiem.
(kernel)
Krótka historia systemu operacyjnego UNIX
Dlaczego UNIX stał się popularny?

- mały, elastyczny, tani, dostępny na komputerach od micro do super komputerów
- właściwości zazwyczaj dostępne tylko w dużych systemach operacyjnych:
- architektura wieloprocesowa, inicjowanie asynchronicznych procesów
- hierarchiczny system plików

1.6 Cele standaryzacji

Budowa systemów tzw. "otwartych", charakteryzujących się:

- przenośnością aplikacji (portability),
- możliwością współpracy oprogramowania działającego na różnych maszynach (interoperability),
- skalowalnością (scalability), t.j. możliwością rozbudowy sprzętowej i rozbudowy aplikacji bez konieczności zmian systemu.

Istota standardu: określenie interfejsu, a nie implementacji.

1.7 Niektóre standardy i instytucje standaryzujące

SVID - standard firmy AT&T - pierwsza próba standardu interfejsu systemu operacyjnego.

POSIX - standard sponsorowany przez Instytut IEEE (Institute of Electrical and Electronics Engineering). Prace nad tym standardem obejmują: interfejs systemu, programy shell, metody testowania zgodności danego systemu ze standardem, pracę w sieci, zagadnienia ochrony i inne.

OSF (Open System Foundation) stowarzyszenie wiodących firm: HP, IBM, DEC i innych, zajmujące się promocją systemów otwartych. Opracowany został standard OSF/Motif. Aktualne prace obejmują między innymi: Distributed Computing Environment - przetwarzanie w środowisku rozproszonym, Distributed Management Environment - centralne zarządzanie sieciami heterogenicznymi.

ISO - International Standard Organization - koordynuje tworzenie i stosowanie standardów w skali międzynarodowej. Wprowadziła siedmiowarstwowy model odniesienia OSI (Open System Interconnection) dotyczący pracy w sieciach komputerowych.

1.8 Cechy systemu UNIX

- System wielodostępny (ang. multiuser): wielu użytkowników pracujących z jednym komputerem, wrażenie samodzielnej pracy
- Praca interakcyjna (interactive): użytkownik wprowadza polecenie z terminala, UNIX wykonuje polecenie, wyświetla wyniki i czeka na nowe polecenie każdy
- System wielozadaniowy (multitasking): użytkownik może uruchomić jednocześnie wiele programów
- Zapewnia bezpieczeństwo (security, privacy): identyfikacja użytkowników, ochrona dostępu do plików i katalogów, ochrona procesów
- Niezależny od urządzeń system we/wy (device independent I/O): każde urządzenie reprezentowane jest przez plik specjalny systemu
- Komunikacja między procesami (ang. interprocess communication): aplikacje mogą wzajemnie się ze sobą komunikować
- System sieciowy (networking): wbudowane jest oprogramowanie umożliwiające pracę w sieci
- Polecenie/Program użytkowy (commands/utilities): polecenie jest programem użytkowym, można samemu je napisać
- Interpretator poleceń (shell): jest wiele programów shell'a, można go zmienić

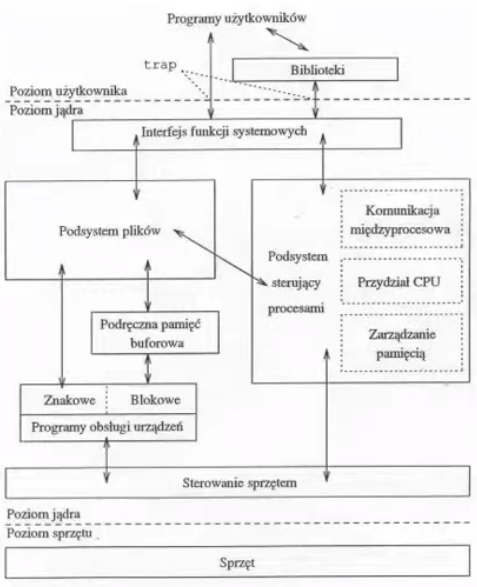
1.9 Struktura systemu operacyjnego

Podsystemy wykonujące zadania:

- Zarządzanie procesami: tworzenie, usuwanie, zawieszanie, odwieszanie procesów, mechanizmy synchronizacji procesów, komunikacja między procesami.
- Zarządzanie pamięcią: zarządzanie pamięcią główną, obszarem wymiany (swap), pojęcie pamięci wirtualnej.
- Zarządzanie przestrzenią dyskową: zarządzanie wolną przestrzenią dysków, procesami, zapisywania informacji na dysku, szeregowanie zadań zapisu i odczytu.
- Zarządzanie operacjami we/wy: obejmuje podsystem buforowania,
- interfejs: urządzenia -sterowniki, sterowniki urządzeń.
- Zarządzanie plikami: tworzenie, usuwanie plików i katalogów, elementarne operacje z plikami i katalogami.
- Podsystem ochrony: ochrona procesów przed działaniem innych procesów, mechanizmy zapewniające, że pliki, segmenty pamięci, CPU, inne zasoby są udostępnione tylko tym procesom, które mają autoryzację systemu operacyjnego. Ogólnie: mechanizmy kontroli dostępu programów, procesów, użytkowników do zasobów systemu komputerowego.
- Praca sieciowa: usługi umożliwiające komunikację w sieci, pojęcie systemów rozproszonych.

1.10 Usługi systemu operacyjnego

- Wykonywanie programów
- Operacje we/wy
- Operacje obsługi systemu plików
- Komunikacja między procesami
- Detekcja błędów
- Przydział zasobów
- Rozliczanie użytkowników (accounting)
- Ochrona
- Funkcje systemowe (system calls):
- interfejs między procesami i systemem operacyjnym



Przykłady funkcji systemowych (system calls):
Podsystem plików: open, read, write, stat
chown, chmod
Podsystem sterowania procesami:
fork, exec, exit, wait, signal

1.11 Pytania

1. Co to jest system operacyjny? Jakie funkcje spełnia?

   System operacyjny to program pośredniczący między sprzętem komputerowym a użytkownikiem. Zadania użytkownika wykonywane są jako procesy obsługiwane przez system operacyjny. System przydziela procesom zasoby komputera: dostęp do jednostki centralnej, pamięć, dyski, i inne.

2. Co to są funkcje systemowe?

   Funkcje systemowe (system call) – stanowią interfejs między wykonywanym programem a (posiadającym zwykle wyższe uprawnienia) jądrem systemu operacyjnego. Funkcje systemowe wywoływane są przez specjalny mechanizm, wspierany przez dany procesor, na przykład z użyciem wyznaczonego przerwania lub instrukcji skoku dalekiego. Podsystem plików: Przyklady:

   - open, read, write, stat (stan)
   - Chown (zmiana właściciela)
   - chmod

   Sterowanie procesami:

   - Fork (utworzenie nowego procesu, nowy proces-klon istniejącego procesu),
   - Exec (umozliwia działanie nowego procesu tak jak chcemy),
   - exit
   - Wait,
   - signal (możliwość wysylania sygnalu do danego procesu)

3. Co jest przedmiotem standaryzacji systemów otwartych?

   Ujednolicenie systemu, języka maszynowego, zasad działania, nazywania, odszukiwania plików, aby dla każdego użytkownika była taka sama. stworzenie kryteriów schematu ich wykonywania. eliminowanie błędów przez co wzrasta efektywność działania systemu. Rozne wersje systemow uzyte w jednolitym systemie komputerowym.

2 Systemy plików

2.1 Pliki i systemy plików w systemie operacyjnym

Pliki to jednostki logiczne przechowywanej informacji, niezależne od właściwości fizycznych urządzeń pamięciowych. Zwykłe w plikach przechowywane są programy lub dane (tekst, liczby, grafika, itp.).

System plików - zbiór typów danych, struktur danych oraz funkcji systemowych (ang. system calls) używanych przez system operacyjny w celu przechowywania informacji w urządzeniach pamięci masowej (głównie na dyskach).

W systemach wielodostępnych systemy plików mają strukturę katalogową (hierarchiczną). Pliki identyfikuje się za pomocą nazw. W systemie UNIX, w zależności od implementacji, pliki mogą mieć krótkie nazwy, do 14 znaków, lub długie nazwy, do 255 znaków.

Zestaw znaków dopuszczalnych obejmuje:

małe lub duże litery, cyfry, znaki specjalne takie jak `,,+, - , _' ."}`
Przykład
Dopuszczalne nazwy plików: `.profile .xyz.abcd abc AbC 123..456..78 -a}`
UWAGA: Tej ostatniej nazwy lepiej jednak nie używać.
UWAGA: W systemie UNIX pliki mogą być identyfikowane za pomocą wielu różnych nazw (dowiązań).

2.2 Podstawowe typy plików

Podstawowe typy plików: pliki zwykłe, pliki specjalne (urządzeń), katalogi, potoki nazwane, gniazda.

- pliki zwykłe,

  W plikach zwykłych przechowujemy programy, dane, teksty, grafikę, itp. W systemie UNIX pliki zwykłe nie mają ustalonego formatu. Plik zwykły jest po prostu ciągiem bajtów o danej długości. Oczywiście aplikacje mogą tworzyć pliki o ściśle ustalonym formacie, na przykład plik assembler generuje plik, w którym wyróżniamy: nagłówek, kod wykonywalny, inicjalizowane dane.

- pliki specjalne,

  Pliki specjalne, nazywane również plikami urządzeń zapewniają łączność z urządzeniami, na przykład z dyskami, terminalami, napędami taśmy, itp. W plikach specjalnych nie przechowuje się żadnych danych. Pliki te charakteryzują sposób działania urządzenia, wskazują miejsce podłączenia urządzeń do systemu oraz zapewniają dostęp do programów obsługi urządzeń („drajwerów").

- katalogi,

  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 pliku znajdującego się w katalogu (lub podkatalogu) oraz pewien numer, pozwalający na odszukanie atrybutów pliku i danych, które się w nim znajdują.

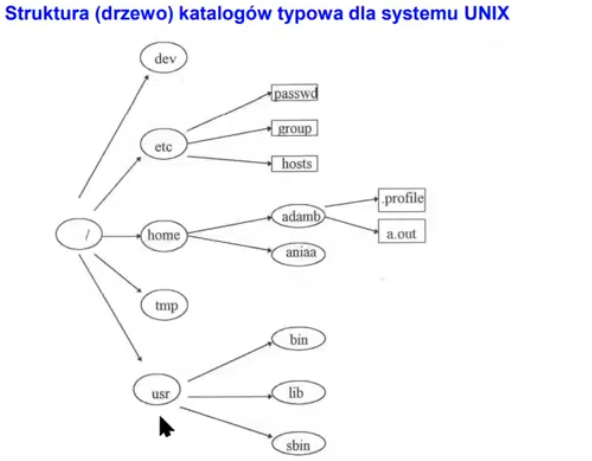
- dowiązania symboliczne - operacja nadania dodatkowej nazwy istniejącemu plikowi,

  Dowiązania zwykłe mogą być tworzone w obrębie tego samego systemu plików. Dowiązań symbolicznych używa się ponad granicami systemów plików oraz w odniesieniu do katalogów. Przechowywane są w nich ścieżki dostępu do plików lub katalogów, na które dowiązania te wskazują.

- potoki nazwane FIFO (ang. named pipe),

  Potoki nazwane (FIFO) wykorzystywane są do komunikacji między procesami. Do tworzenia tych potoków wykorzystywane są odpowiednie procedury biblioteczne. Procesy mogą otwierać potoki nazwane do odczytu i zapisu, tak jak otwierają pliki zwykłe. Gniazda wprowadzone zostały w systemie BSD UNIX. Są wykorzystywane również do komunikacji między procesami. Wykorzystują jednak inne mechanizmy niż potoki nazwane.

- gniazda (ang. UNIX–domain sockets).



2.3 Ścieżka dostępu

Struktura systemu plików w systemie UNIX.

Jeden z katalogów zawsze służy użytkownikowi pracującemu w systemie UNIX jako katalog bieżący.

Położenie pliku lub katalogu w drzewie katalogów określa ścieżka dostępu do pliku lub katalogu. Bezwzględna ścieżka dostępu określa położenie pliku lub katalogu względem katalogu głównego (ang. root directory) , na przykład:

```
/etc/passwd
/home/adamb/.profile
```

Względna ścieżka dostępu określa położenie pliku lub katalogu względem katalogu bieżącego, na przykład (jeśli katalogiem bieżącym jest `/home`):

```
adamb
adamb/a.out
../usr/lib
```

2.4 Operacje dotyczące katalogów

- zmiana katalogu bieżącego, `cd [ścieżka dostępu do katalogu]`
- ustalenie nazwy katalogu bieżącego, `pwd`
- sprawdzenie zawartości katalogu, `ls [-opcje] [ścieżka dostępu do katalogu]`
- tworzenie katalogów, `mkdir [-opcje] [ścieżka dostępu do katalogu]`
- usuwanie katalogów `rmdir [-opcje] [ścieżka dostępu do katalogu]`
- utworzenie dowiązania do katalogu: `ln -s stara_nazwa nowa_nazwa`

2.5 Operacje odnoszące się do plików

- wypisanie zawartości pliku tekstowego: `cat [ścieżka dostępu do pliku]}, \verb`more [ścieżka dostępu do pliku]
- drukowanie pliku: `lp [-opcje] [ścieżka dostępu do pliku]}`
- wypisanie atrybutów pliku: `ls [-opcje] [ścieżka dostępu do pliku]}`
- kopiowanie pliku: `cp [-opcje] co_kopiujemy dokąd_kopiujemy}`
- usuwanie pliku: `rm [-opcje] nazwa_pliku_lub_katalogu}`
- zmiana nazwy pliku lub jego przeniesienie: `mv [-opcje] stara_nazwa nowa_nazwa}`
- utworzenie dowiązania do pliku lub katalogu: `ln [-opcje] stara_nazwa nowa_nazwa}`

2.6 Atrybuty plików i katalogów

- typ pliku,
- prawa dostępu do pliku,
- liczba dowiązań do pliku,
- identyfikator właściciela,
- identyfikator grupy,
- rozmiar pliku w bajtach,
- czas ostatniej modyfikacji pliku,
- czas ostatniego dostępu do pliku,
- czas ostatniej zmiany informacji w i-węźle,
- nazwa pliku.

Przykład:

```
\$ ls -ld /etc
drwxr-xr-x 22 root root 1024 Sep 1995 /etc
```

2.7 Pojęcie i-węzła

Każdemu plikowi przyporządkowany jest i-węzeł (ang. i.node), który jest rekordem przechowującym większość informacji o pliku.

Zawartość i-węzła:

- typ pliku,
- prawa dostępu do pliku,
- liczba dowiązań do pliku,

- identyfikator właściciela,
- identyfikator grupy,
- rozmiar pliku w bajtach,
- czas ostatniej modyfikacji pliku,
- czas ostatniego dostępu do pliku,
- czas ostatniej zmiany informacji w i.węźle,
- 12 wskaźników zawierających adresy bloków z danymi pliku (bloki bezpośrednio adresowane),
- wskaźnik zawierający adres bloku, w którym przechowywane są adresy bloków z danymi (adresowanie pośrednie jednostopniowe),
- wskaźnik zawierający adresy bloków, w których przechowywane są adresy bloków z adresami bloków z danymi (adresowanie pośrednie dwustopniowe),
- wskaźnik wykorzystywany w adresowaniu pośrednim trzystopniowym.

i-węzły są tworzone wtedy, gdy tworzony jest system plików. Liczba i-węzłów w systemie plików zależy od jego rozmiaru oraz założonego średniego rozmiaru pliku (np. 2kB lub 6kB).

Każdy i-węzeł zajmuje 128 bajtów. i-węzły tworzą tablicę i-węzłów.

Poszczególne i-węzły identyfikowane są przez numery, określające ich położenie w tablicy i-węzłów.

Aby sprawdzić, jaki i-węzeł został przyporządkowany danemu plikowi, należy w poleceniu `ls}` użyć opcji
`-i}`.

Przykład:

```
$ ls -lid /etc
77 drwxr-xr-x 22 root root 1024 Sep 1995 /etc
```

Nazwy plików są przechowywane w katalogach, łącznie z numerami odpowiadających tym plikom i-węzłów. Dzięki temu możliwe jest odczytanie atrybutów pliku oraz odszukanie przechowywanych w nim danych.

2.8 Zawartość i-węzła

Blok identyfikacyjny (ang. Superblock) zawiera między innymi:

1. rozmiar systemu plików,
2. liczbę wolnych bloków w systemie plików,
3. listę wolnych bloków dostępnych w systemie plików,
4. indeks następnego wolnego bloku na liście wolnych bloków,
5. rozmiar tablicy i-węzłów,
6. liczbę wolnych i-węzłów w systemie plików,
7. listę wolnych i-węzłów w systemie plików,
8. indeks następnego wolnego i-węzła na liście wolnych i-węzłów.W

2.9 Adresowanie bloków danych, Sposób przechowywania plików na dysku

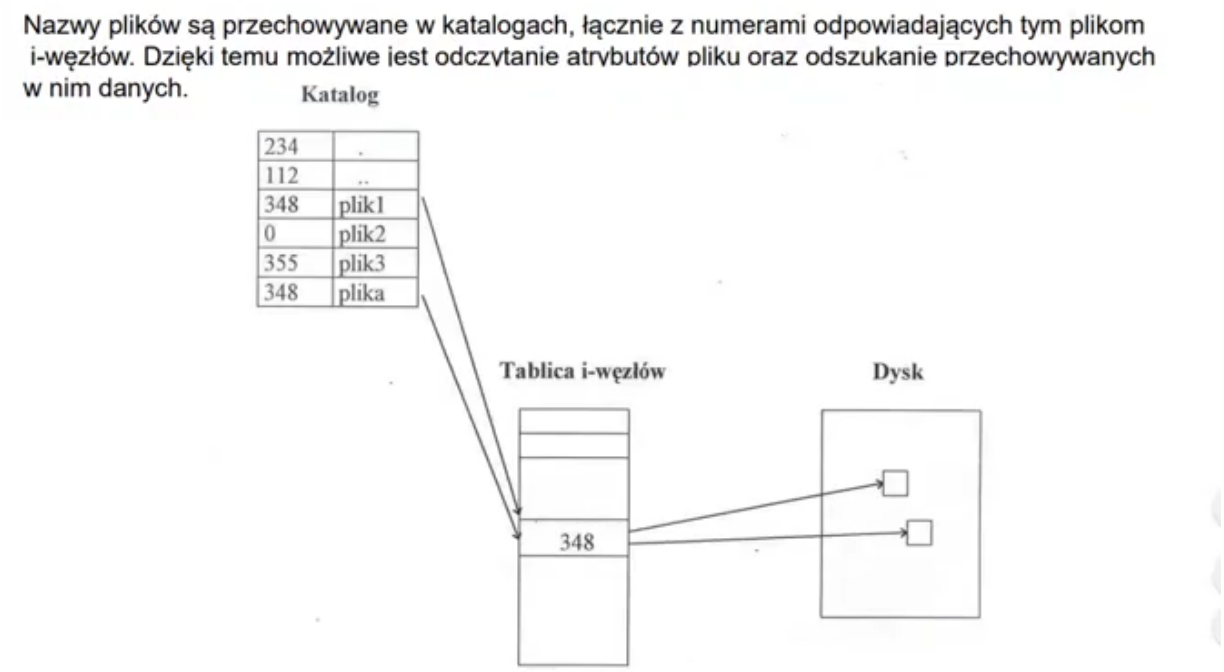
Adresowanie tylko bezpośrednie nie jest efektywne i znacznie ograniczyłoby rozmiary plików.

W systemie UNIX zastosowano następujące reguły:

- tablica adresów przechowywana w i-węźle ma 15 elementów (wskaźników) i każdy zajmuje 4 bajty,
- 12 pierwszych wskaźników zawiera adresy bloków z danymi,
- następny, 13 wskaźnik zawiera adres bloku, w którym znajdują się adresy bloków z danymi,
- 14 wskaźnik to adres bloku, w którym umieszczane są adresy bloków zawierających adresy bloków z danymi,
- 15 wskaźnik to adres bloku, w którym umieszczane są adresy bloków przeznaczonych na adresy następnych bloków zawierających adresy bloków z danymi.

Przykład: jeśli blok zajmuje 4 kB, to

- adresowanie bezpośrednie pozwala na zaadresowanie danych plików o rozmiarach nie przekraczających: 48 kB,
- adresowanie pośrednie pozwala na zaadresowanie danych plików o rozmiarach nie przekraczających: 48 kB + 1024*4 kB = 4144 kB,
- podwójne adresowanie pośrednie pozwala na zaadresowanie danych plików o rozmiarach nie przekraczających: 48 kB + 1024*4 kB + 1024*1024*4 kB = 4198448 kB,
- adresowanie pośrednie pozwala na zaadresowanie danych plików o rozmiarach nie przekraczających: 48 kB + 1024*4 kB + 1024*1024*4 kB + 1024*1024*1024*4 kB,



2.10 Zadania

Atrybuty plików Polecenie ll (alias polecenia ls –l) pokazuje miedzy innymi czas ostatniej modyfikacji pliku. Polecenie to z opcją –u pokazuje czas ostatniego dostępu do pliku, natomiast z opcją -c pokazuje czas ostatniej zmiany informacji przechowywanych w i-węźle.

O godz. 19:00 poleceniem

```
$ /usr/bin/date > abc
```

utworzono plik abc, po czym sprawdzono atrybuty tego pliku:

```
$ ll abc
-rw-r--r-- 1 user1 users 29 Sep 13 19:00 abc
```

Następnie o godz. 19:10 wydano polecenie:

```
$ /usr/bin/date >> abc
```

Podaj dokładnie co pokażą poniższe polecenia i uzasadnij odpowiedź:

```
$ ll abc
$ ll -u abc
$ ll -c abc
```

O godz. 19:20 wydano polecenie:

```
$ ln abc xyz
```

Podaj dokładnie co pokażą poniższe polecenia i uzasadnij odpowiedź:

```
$ ll abc
$ ll -u abc
$ ll -c abc
```

O godz. 19:25 wydano polecenie:

```
$ chmod +x abc
```

Podaj dokładnie co pokażą poniższe polecenia i uzasadnij odpowiedź:

```
$ ll xyz
$ ll -u xyz
$ ll -c xyz
```

2.11 Rozwiązanie

Atrybuty plików Polecenie ll (alias polecenia ls –l) pokazuje miedzy innymi czas ostatniej modyfikacji pliku. Polecenie to z opcją –u pokazuje czas ostatniego dostępu do pliku, natomiast z opcją -c pokazuje czas ostatniej zmiany informacji przechowywanych w i-węźle.

O godz. 19:00 poleceniem

```
$ /usr/bin/date > abc
```

utworzono plik abc, po czym sprawdzono atrybuty tego pliku:

```
$ ll abc
-rw-r--r-- 1 user1 users 29 Sep 13 19:00 abc
```

Następnie o godz. 19:10 wydano polecenie:

```
$ /usr/bin/date >> abc
```

Podaj dokładnie co pokażą poniższe polecenia i uzasadnij odpowiedź: Zmieniła się wielkość pliku dwukrotnie, ponieważ dopisujemy do pliku kolejną informację co powiększa Jego rozmiar, w każdym z przypadków.

```
$ ll abc
-rw-r--r-- 1 user1 users 58 Sep 13 19:10 abc
```

W tym przypadku czas się zmienia jako czas modyfikacji.

```
$ ll -u abc
-rw-r--r-- 1 user1 users 58 Sep 13 19:00 abc
```

Tutaj czas odczytu się nie zmienia, czyli to czas utworzenia.

```
$ ll -c abc
-rw-r--r-- 1 user1 users 58 Sep 13 19:10 abc
```

Tutaj zmienia się czas, ponieważ przez modyfikację rozmiaru zmienia się także informacja o i-węźle.

O godz. 19:20 wydano polecenie:

```
$ ln abc xyz
```

Podaj dokładnie co pokażą poniższe polecenia i uzasadnij odpowiedź:

Zmienia się liczba dowiązań na 2 oraz godzina na i-węźle

```
$ ll abc
-rw-r--r-- 2 user1 users 58 Sep 13 19:10 abc
```

Czas modyfikacji się nie zmienia.

```
$ ll -u abc
-rw-r--r-- 2 user1 users 58 Sep 13 19:00 abc
```

Czas dostępu również się nie zmienia.

```
$ ll -c abc
-rw-r--r-- 2 user1 users 58 Sep 13 19:20 abc
```

Czas na i-węźle się zmienia, ponieważ utworzono dowiązanie.
O godz. 19:25 wydano polecenie:

```
$ chmod +x abc
```

Podaj dokładnie co pokażą poniższe polecenia i uzasadnij odpowiedź: Zmieniają się uprawnienia dostępu, czyli plik staje się wykonywalny

```
$ ll xyz
Rwxr -xr { x 2 user1 users 58 Sep 13 19:10 xyz*
```

Zmienia się nazwa pliku, czas modyfikacji pozostaje bez zmian

```
$ ll -u xyz
Rwxr -xr { x 2 user1 users 58 Sep 13 19:00 xyz*
```

Zmienia się nazwa pliku, czas odczytu pozostaje bez zmian

```
$ ll -c xyz
Rwxr -xr { x 2 user1 users 58 Sep 13 19:25 xyz*
```

Zmienia się nazwa pliku i czas na i-węźle, ponieważ zmieniliśmy uprawnienia do pliku.

3 Sposób przechowywania plików na dysku

Do przechowywania danych na dysku system UNIX używa bloków i fragmentów. Fragment jest najmniejszą jednostką przestrzeni dyskowej zajmowanej przez plik. Rozmiar bloku jest całkowitą wielokrotnością rozmiaru fragmentu (stosunek rozmiaru bloku do rozmiaru fragmentu nie może jednak przekraczać 8):
np. rozmiar bloku wynosi 8 kB a rozmiar fragmentu 2kB.

3.1 Reguły przydzielania bloków i fragmentów

1. Jeśli rozmiar pliku jest mniejszy niż rozmiar fragmentu, plikowi temu przydzielany jest pierwszy wolny fragment.

2. Jeśli rozmiar pliku jest większy niż rozmiar fragmentu, ale mniejszy niż rozmiar bloku, plikowi temu przydzielane są kolejne fragmenty należące do tego samego bloku.

3. Jeśli rozmiar pliku Jest większy niż rozmiar bloku, to plikowi przydzielana jest odpowiednia liczba bloków, niekoniecznie znajdujących się obok siebie, o łącznym rozmiarze nie przekraczającym rozmiaru pliku.

Pozostała część pliku umieszczana jest zgodnie z regułami 1 oraz 2.

Do przechowywania informacji o stanie wolnych bloków i fragmentów można wykorzystać mapę bitową, na przykład dla bloków 8 kB i fragmentów 2 kB :

```
1110    1110    0100    0011    0000    0001

blok 0  blok 1  blok 2  blok 3  blok 4  blok 5  ... -.
```

0 - fragment wolny
1 - fragment zajęty
Kolejność adresów w i-węźle jest zgodna z kolejnością informacji w pliku!
W bloku którego adres jest jako pierwszy, będzie pierwsza, początkowa część informacji.

3.2 Struktura systemu plików

Blok identyfikacyjny Tablica i-węzłów Bloki danych

Podział dysków na partycje Dyski bywają dzielone na sekcje fizyczne. Taki podział ma szereg wad. Do najważniejszych należą:

- nieefektywne wykorzystanie miejsca na dysku,
- rozmiar sekcji (a w wyniku rozmiar systemu plików nie większy niż rozmiar dysku),
- nie można zmienić rozmiarów sekcji (systemu plików)

Niektóre implementacje systemu UNIX wykorzystują partycje logiczne LVM

Każda partycja (sekcja) jest traktowana jako niezależny wirtualny dysk. Jest reprezentowana przez plik specjalny (urządzenia), podobnie jak każdy dysk

W partycjach tworzone są systemy plików. Można je również wykorzystać jako obszary wymiany (swap). Systemy plików utworzone w poszczególnych partycjach muszą być dowiązane do głównego systemu plików:

```
mount plik_specjalny_reprezentujący_partycję katalog
```

Operacja odwrotna:

```
umount plik_specjalny_reprezyntujący_partycję
umount katalog
```

Mechanizm dysku z ruchomymi głowicami
Różne sposoby przydziału miejsca na dysku

- Przydział ciągły
- Przydział listowy
- Przydział indeksowy

Metody dostępu do informacji pliku

- Dostęp sekwencyjny
- Dostęp bezpośredni (swobodny)

Ilustracja ciągłego przydziału miejsca na dysku
Ilustracja listowego przydziału miejsca na dysku
Tablica przydziału plików (File Allocation Table)
Ilustracja indeksowego przydziału miejsca na dysku

4 Zarządzanie procesami

4.1 Pojęcie procesu

4.2 Tworzenie, usuwanie, zawieszanie i odwieszanie procesów.

4.3 Komunikacja między procesami.

4.4 Szeregowanie procesów.

5 Zarządzanie pamięcią

5.1 Pamięć główna

5.2 Pamięć pomocnicza na dysku (obszar wymiany „swap")

5.3 Pojęcie pamięci wirtualnej

5.4 Procesy wymiany

5.5 Procesy stronicowania i segmentacji

6 Zarządzania operacjami wejścia wyjścia

Podstawowe pojęcia (szyna, port we-wy, sterownik, odpytywanie, system przerwań)
Struktura oprogramowania we-wy w jądrze SO
Urządzenia znakowe, urządzenia blokowe
Pliki specjalne
Tablica rozdzielcza urządzeń

7 Laboratorium

7.1 Podstawy użytkowania systemu

Praca w systemie.
Postać poleceń w systemie UNIX .

7.2 Wybrane polecenia podstawowe

Polecenia służące identyfikacji użytkowników w systemie oraz komunikacji między użytkownikami.

7.3 System plików

Poruszanie się w systemie plików.
Tworzenie i usuwanie katalogów.

7.4 Praca z plikami

Plik i jego atrybuty .
Działania na plikach.
Dowiązywanie plików i katalogów.

7.5 Prawa dostępu do plików i katalogów

Podstawowe prawa dostępu
Zmiana praw dostępu.

7.6 Podstawy edycji tekstów

Edytory w systemie UNIX Praca z edytorem vi

7.7 Podstawy korzystania z shella

Rodzaje shella
Wykonywanie poleceń w shellu
Środowisko shella

7.8 Środowisko użytkownika

Określanie środowiska użytkownika. Zmienne shella

7.9 Różne mechanizmy w shellu

Generowanie nazw plików. Przeadresowanie wejścia i wyjść

7.10 Potoki

Podstawianie poleceń Korzystanie z wcześniej wydanych poleceń

8 Wybrane polecenia działania na plikach tekstowych

8.1 Nadzorowanie wykonywania procesów

Informacja o procesach Przetwarzanie w pierwszym i drugim planie Wysyłanie sygnałów do procesów Planowanie wykonywania poleceń

8.2 Wybrane polecenia użytkowe związane z archiwizacją plików

Kompresja danych Wyszukiwanie plików Tworzenie kopii zapasowych

8.3 Wprowadzenie do pracy w sieciach komputerowych

Sieci komputerowe, sieć INTERNET Poczta elektroniczna Praca na zdalnym komputerze

9 Literatura

A. Silberschatz, P.B. Galvin, G. Gaigne; Podstawy systemów operacyjnych, WNT, Warszawa 2005.

M. J. Bach; Budowa systemu operacyjnego UNIX, WNT, Warszawa, 1995.

B. Goodheart, J. Cox; Sekrety magicznego ogrodu, Unix system V wersja 4 od środka, WNT, Warszawa, 2001.

B. Goodheart, J. Cox; Sekrety magicznego ogrodu, Unix system V wersja 4 od środka, - klucz do zadań. WNT, Warszawa, 2001.

K. Haviland, D. Gray, B. Salama; UNIX. Programowanie systemowe. Wydawnictwo RM, Warszawa, 1999.

A. Silberschatz, P.B. Galvin, G. Gaign; Operating System Concepts. John Wley & Sons, Inc., 2003.

A. S. Tanenbaum; Modern Operating Systems, Prentice-Hall, Inc. London, 1992.

E. Nemeth, G. Snyder, S. Seebass, T.R. Hein: Przewodnik administratora systemu UNIX. WNT, Warszawa, 1998.

S. Prata, D. Martin; Biblia systemu UNIX V. Polecenia i programy użytkowe, LT&P, Warszawa, 1994.

A. Southerton, E.C. Perkins; Słownik poleceń systemów UNIX i X, Wiley & WNT, Warszawa , 1995.

L. Lamb; Learning the vi editor. O'Reilly & Associates, Inc., Sebastopol, CA, 1996.

M.I. Bolsky, D.G. Korn; The Kornshell - command and programming language, Prentice Hall, 1989.