System operacyjny Unix;
administrowanie

- pliki konfiguracyjne Unix-a z katalogu /etc
  - plik "/etc/passwd";

zawiera linie następującej postaci:

name : passwd : uid : gid : gecos : home_dir : shell

początkowy shell

początkowa grupa użytkowników (id grupy)

zamiast hasła może być "x" co oznacza ze hasło jest w pliku shadow

przykład:

/etc/passwd

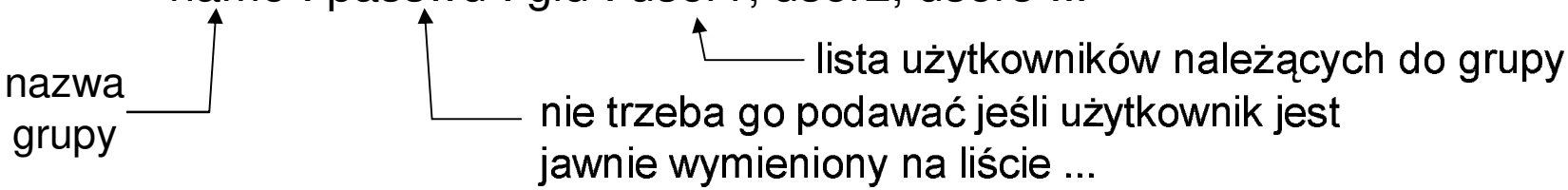
```
user1:x:502:503::/home/user1:/bin/bash
user2:x:503:504::/home/user2:/bin/bash
user3:x:504:505::/home/user3:/bin/bash
```

/etc/shadow

```
user1:$1$zIbhRrsb$LvbSB4fYWEQ1YrxRNWCxY.:12386:0:99999:7:::
user2:$1$nEM0IHuV$SdJR98gE.EHMPVfCYMuDc0:12386:0:99999:7:::
user3:$1$CXtZnR6B$V8W5KbGwcrGA6mX9XJQlQ.:12386:0:99999:7:::
```

System operacyjny Unix;
administrowanie

- pliki konfiguracyjne Unix-a z katalogu /etc
  - plik "/etc/group";
    zawiera linie następującej postaci:
    name : passwd : gid : user1, user2, user3 ...



  - polecenia związane z grupami:
    - groups – wyswietla wszystkie grupy do których należymy
    - newgrp *nazwa_grupy* – zmienia bieżącą grupę (być może zostaniemy zapytani o hasło !)
    - chgrp – zmiana grupy właścicieli pliku
    - gropupadd – dodawanie nowej grupy (tylko *root;* użytkowników grupy dopisywać ręcznie)
    - gpasswd – zmiana hasła do grupy
    - id – wyświetla uid, gid, oraz grupy do których należymy
  - różnica w działaniu grup użytkowników w SV i BSD:
    - SysV: w danej chwili użytkownik należy do jednej grupy
    - BSD: w danej chwili użytkownik ma prawa przysługujące wszystkim grupom do których należy

System operacyjny Unix;
administrowanie

- pliki konfiguracyjne Unix-a z katalogu /etc
  - plik "/etc/group" – eksperyment
    - mamy użytkowników: *user1*, *user2*, *user3*
    - mamy grupy użytkowników:
      - *test1i2* zawierającą użytkowników *user1* i *user2*
      - *test3* zawierającą użytkownika *user3*
      - *test1i3* zawierającą użytkowników *user1* i *user3*
    - mamy pliki

```
-rw-rw-r--   1 mhanckow test1i2    16 maj 31 08:41 plik1.txt
-rw-rw-r--   1 mhanckow test1i2     1 maj 31 08:19 plik2.txt
-rw-rw-r--   1 mhanckow test3       1 maj 31 08:21 plik3.txt
-rw-rw-r--   1 mhanckow test1i3     1 maj 31 08:21 plik4.txt
```

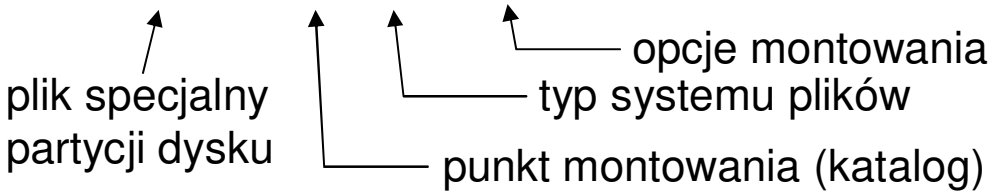
    - wejść do systemu jako *user1* i zobaczyć jakie operacje możemy wykonywać na plikach plik?.txt ...
    - przełączyć się na inną grupę poleceniem *newgrp* (listę bieżących grup wyświetlimy poleceniem *id*)
    - grupy linux-owe działają jak w BSD ?!?!?! (= mamy dostęp wynikający ze wszystkich grup do których należymy wyświetlanych przez *id*)
    - można wchodzić do grup do których "nie należymy" podając hasło !!!

System operacyjny Unix;
administrowanie

- pliki konfiguracyjne Unix-a z katalogu /etc
  - plik "/etc/fstab";

    zawiera systemy plików montowane podczas uruchamiania;
    każda linia wygląda jak argumenty polecenia "mount"

```
/dev/hda3 /mnt/hda3 ntfs exec,dev,suid,ro,uid=501,gid=501 1 1
/dev/hda1 / ext2 noatime 1 1
```



System operacyjny Unix;
administrowanie

- pliki konfiguracyjne Unix-a z katalogu /etc
  - plik "/etc/inittab";
    - plik używany podczas zmiany *poziomu działania* [ang. runlevel]
    - określa które procesy (demony) mają być uruchomiona na danym poz iomie
    - plik inittab jest interpretowany przez proces *init* (pid=1 !)
  - każda linia tego pliku wygląda tak:

    id : run levels : action : command

```
np:
    l1:1:wait:/etc/rc.d/rc 1
    l2:2:wait:/etc/rc.d/rc 2
    l3:3:wait:/etc/rc.d/rc 3
    1:2345:respawn:/sbin/mingetty tty1
    2:2345:respawn:/sbin/mingetty tty2
    3:2345:respawn:/sbin/mingetty tty3
```

skrypty uruchamiane przy przełączaniu na dany poziom działania (np. podczas startu systemu)

  - poziomy działania:

    1 – single user mode (tryb do wykonywania czynności adm)

    2-5 – normalne działanie

    6 – reboot
  - zmiana poziomu działanie: polecenie "telinit poziom"

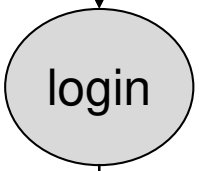
... pokazać plik /etc/inittab Linuxa

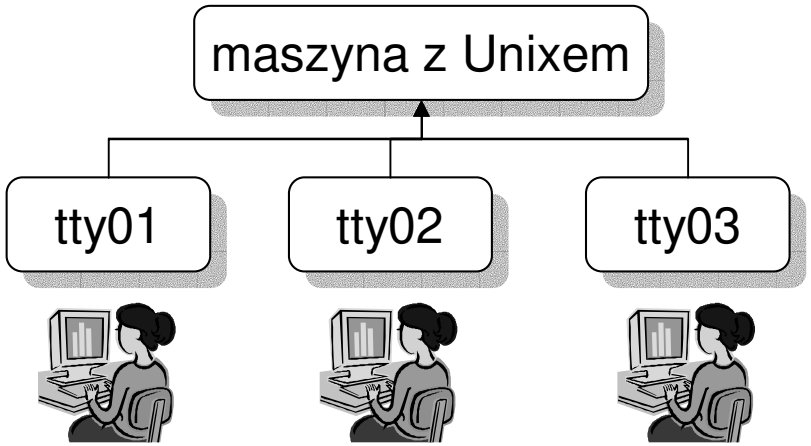
System operacyjny Unix;
administrowanie

- jak użytkownicy są wpuszczani do systemu
  - mamy na myśli użytkowników korzystających z systemu uniksowego poprzez terminale tekstowe ... (Linux: Ctrl+Alt+F?)
  - proces *init* uruchamia procesy *getty* dla każdego terminala

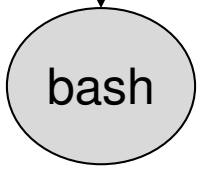
uruchamiany przez proces init
na podstawie pliku /etc/inittab;
dla każdego terminala osobny
proces getty

gdy użytkownik
naciśnie klawisz
na terminalu tty01
zostaje włączony login
(fun. sys. exec())





pyta o id i hasło;
sprawdza w pliku /etc/passwd;
jeśli ok. to uruchamia powłokę
i ustawia uid oraz gid



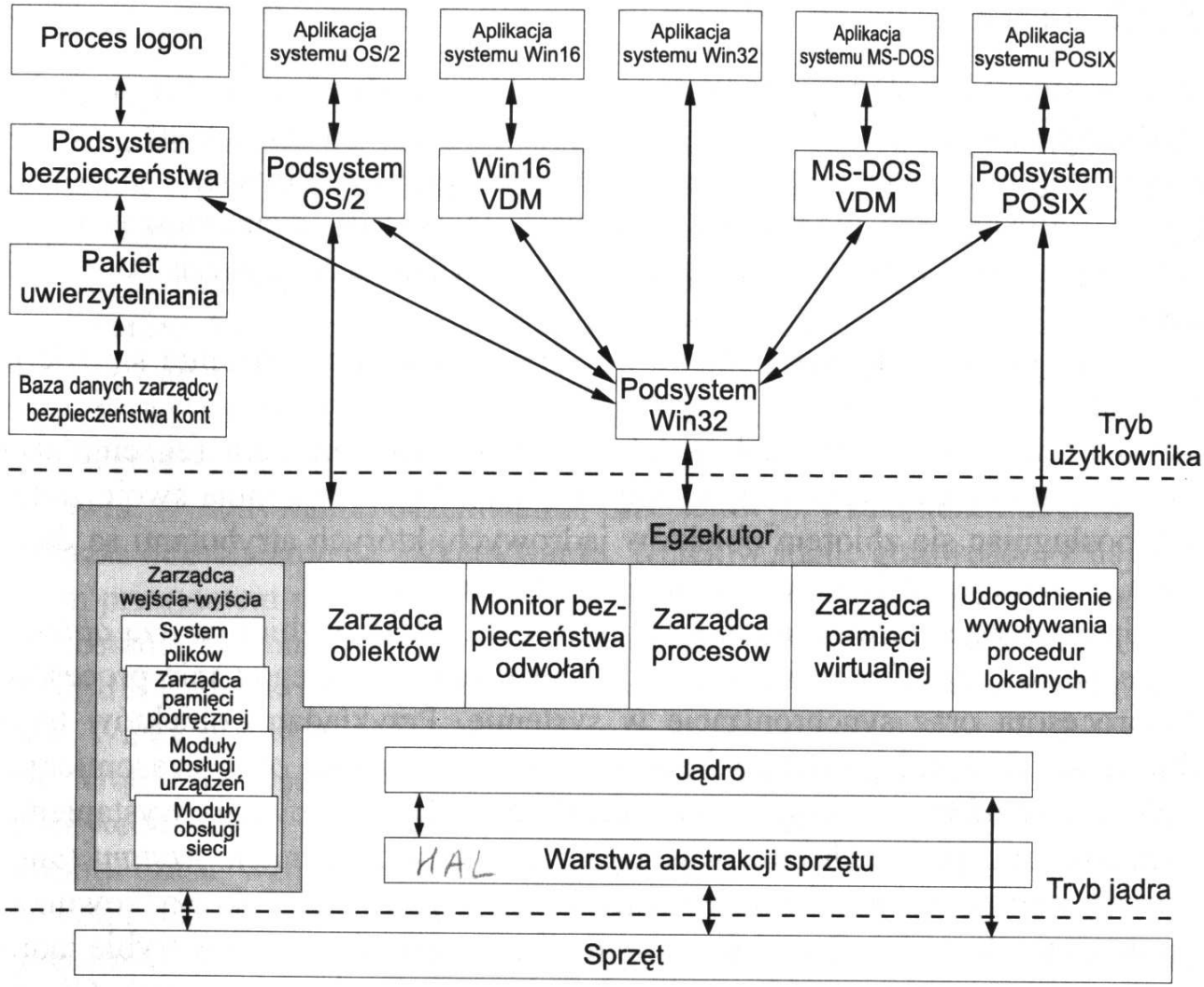
Windows NT 4.0
Windows 2000
Windows XP

System operacyjny WinNT

- cechy systemu
  - wieloprogramowy, podział czasu, procesy i "wątki poziomu jądra"
  - jedno-dostępny (!?!?!?!?!)
  - "przenośny" na różne komputery (z procesorem Intel-a, DEC Alpha)
  - może działać na komputerach z wieloma procesorami
  - można uruchamiać programy starszych s.o. firmy Microsoft (DOS, WIN16, OS/2)
  - w WinNT 4.0 GUI wzięte z Win95
  - możliwość kompilowania programów POSIX (programy używające tradycyjnych fun. sys. Unix-a !)
  - system plików NTFS,
    doskonalszy od FAT; odporny na uszkodzenia bo wszelkie operacje na strukturach danych s. plików są robione przy pomocy transakcji (mechanizm z b.d. zapewniający spójność danych nawet po awarii ...)
  - prawa dostępu do plików/katalogów w NTFS na zasadzie ACL (ang. Access Control List) – każdy plik/katalog ma listę użytkowników i ich prawa
  - wbudowana obsługa sieci (tzw sieć Microsoft); możliwość udostępniania zasobów w sieci (zasobów takich jak katalogi, drukarki)

System operacyjny WinNT

- architektura Win NT 4.0 ...



Rys. 23.1 Schemat blokowy systemu Windows NT

System operacyjny WinNT

- architektura Win 2000 ...

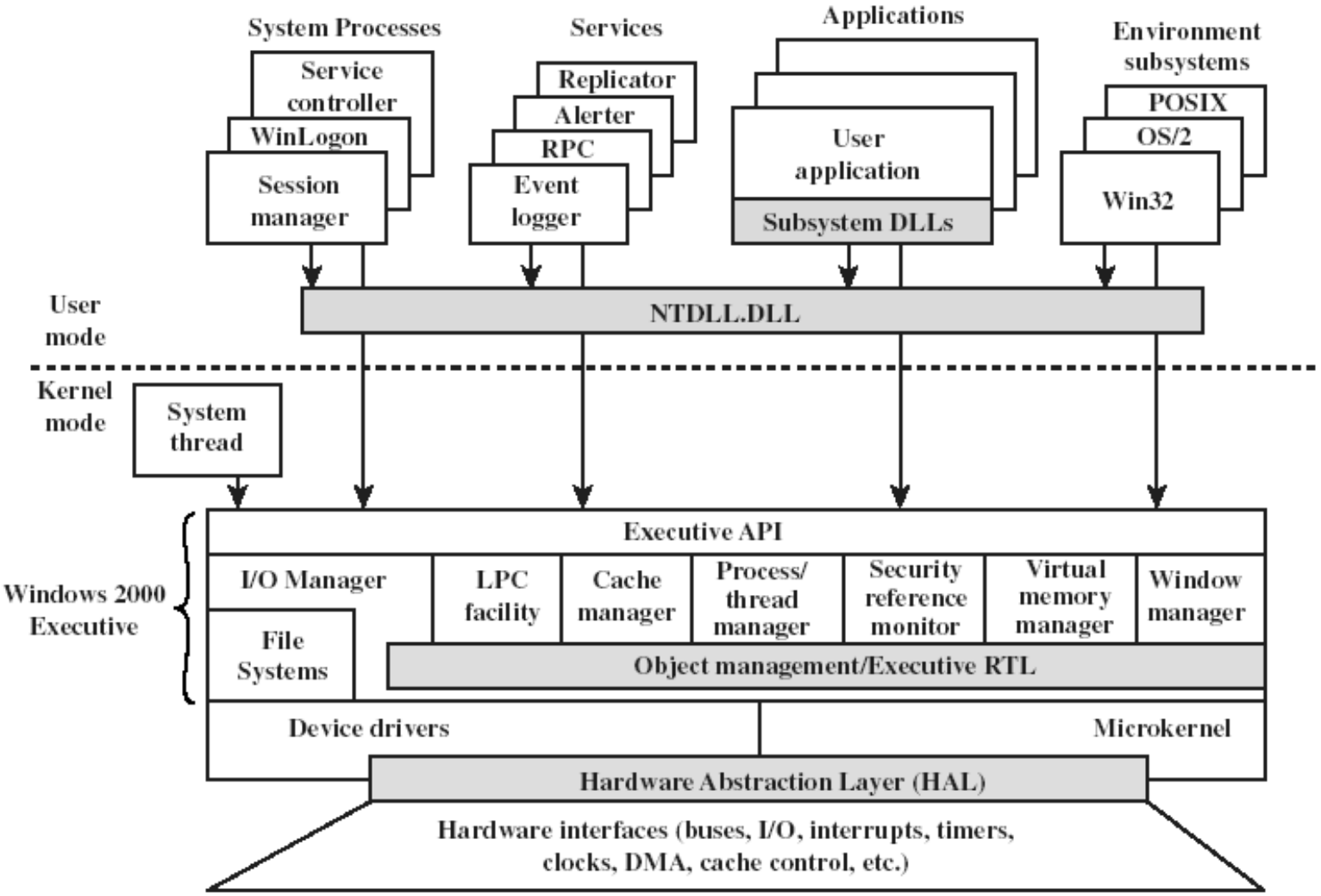


Figure 2.13 Windows 2000 Architecture

System operacyjny WinNT

- architektura
  - zasadnicza cecha: oparta na mikrojądrze
    (ale "zmodyfikowana" – procesy usługowe (=procesy serwery) działaja w trybie jądra a nie w trybie użytkownika jak w prawdziwej "arch z mikrojądrem")
  - mikrojądro zajmuje się:
    - przełączaniem procesora między wątkami i procesami (+planowanie)
    - obsługą wyjątków i przerwań sprzętowych
    - jedynie mikrojądro NIE działa jako proces/ wątek
    - dzięki arch z mikrojądrem łatwo wprowadzać nowe podsystemy/ procesy usługowe bez "rewolucji" w reszcie systemu ...
  - specyficzne nazewnictwo:
    - egzekutor = jądro; zawiera procesy usługowe
    - procesy usługowe (ang. executive services) – w nich jest realizowana większość usług s.o.; inna nazwa "zarządcy"
    - podsystemy środowiskowe (ang. environment subsystems) – w nich stwarza się środowisko rożnych s.o. (DOS, WIN16, POSIX, Win32);
    - Win32 to główny ("wrodzony") podsystem środowiskowy
    - podsystemy środowiskowe i procesy usługowe są normalnymi procesami !!! (różnica: podsystemy działają w trybie użytkownika, proc usługowe w trybie jądra – tj są bardziej uprzywilejowane)

System operacyjny WinNT

- architektura c.d.
  - są ładowalne moduły obsługi urządzeń (ang. drivers);
    dzięki temu system może zostać uzupełniony o:
    - nowe urządzenia we/wy (np. nowa karta graficzna)
    - nowe systemy plików (np. NT4.0 nie obsługuje FAT32 ale można dodać driver)
    - nowe rodzaje sieci
  - NT napisano głownie w języku C/C++
    (jednak kod źródłowy nie jest znany publicznie jak w przypadku Linux-a)
  - warstwa abstrakcji sprzętu HAL (ang. Hardware Abstraction Layer);
    tworzy "logiczną maszynę"; maszyny z procesorami Intela i DEC Alpha wyglądają tak samo dzięki temu
  - niektóre procesy usługowe pomijają HAL i odwołują się bezpośrednio do sprzętu (np. moduł graficzny; chodzi o szybkość operacji graficznych)
  - niezawodność:
    - ochrona sprzętowa pamięci (wirtualnej prz. adr. procesu)
    - NTFS – odporność na uszkodzenia dzięki mechanizmowi transakcji
  - LPC/RPC – podsystemy i procesy usługowe jądra komunikują się przy pomocy mechanizmu *zdalnego wywołania procedury* (mechanizm LPC dotyczy wywołań w na jednej maszynie, RPC między różnymi maszynami; RPC jest używane do realizacji przetwarzania rozproszonego ...)

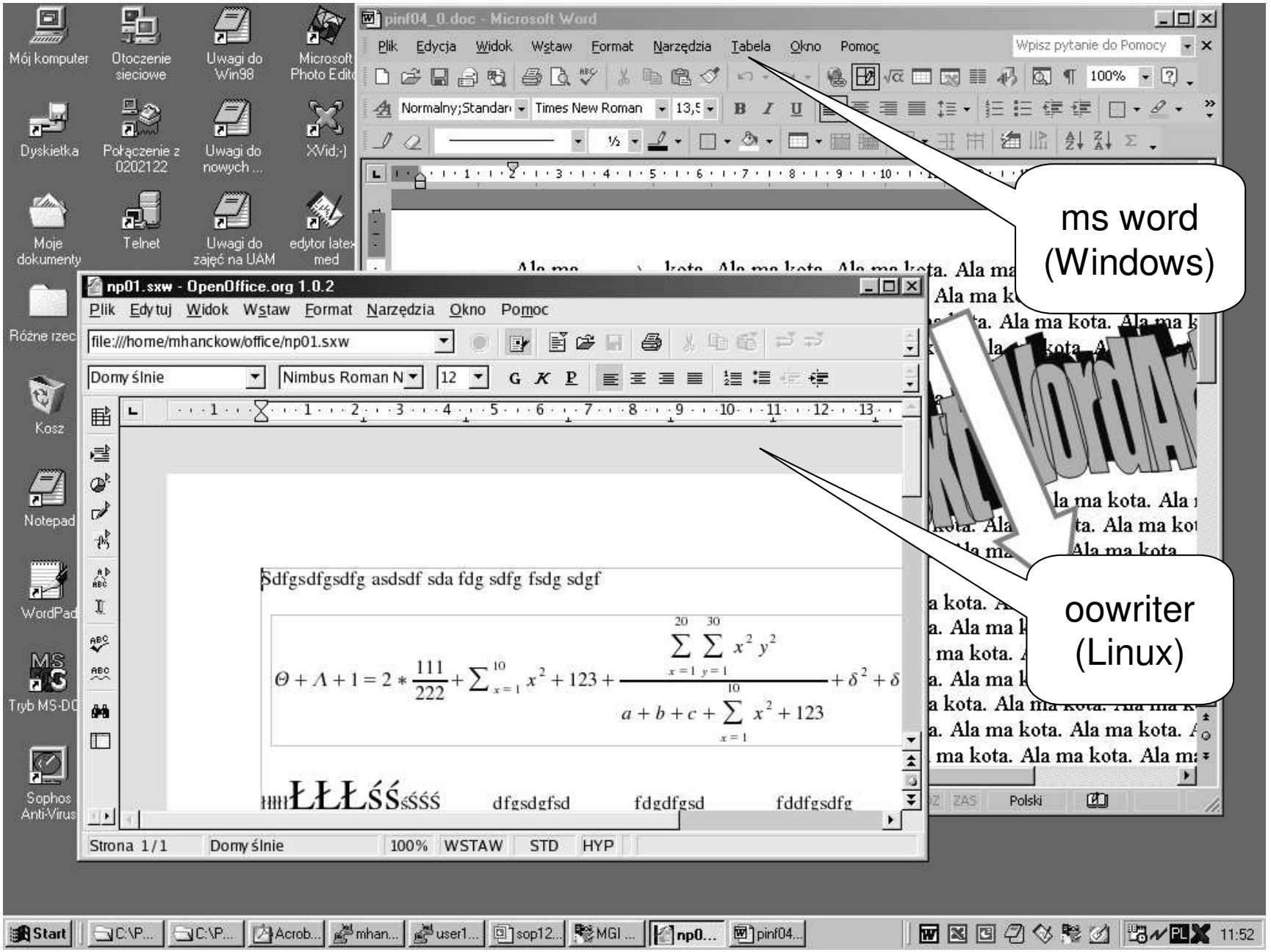
System operacyjny WinNT

- architektura c.d.
  - **rodzaje aplikacji Win32**
    - GUI = Graphical User Interface - otwieraja okna, używają grafiki
    - CUI = Console User Interface – wyglądają jak programy dos-owe (ale nimi nie są! są programami używającymi Win32)
  - **obiekty jądra**
    - każdy elementy s.o. (proces, wątek, semafor, otwarty plik, łączekomunikacjyne, skrzynka pocztowa) jest reprezentowany przez "obiekt jądra" ...
    - obiekty są dostępne poprzez "uchwyty"; poprzez uchwyt do obiektu można na tym obiekcie wykonywać różne operacje
    - programy użytkowników też mogą tworzyć/otwierać obiekty jądra przy pomocy fun. sys. Create*() o Open*()
    - każdy proces posiada "tablice obiektów" (z uchwytami do tych obiektów)
    - przykłady programów Win32:
      pliki np??.cpp + np??.exe
      (omówić ...)

Różne ciekawostki ...

Zdalna praca z Unix-em

- pokażemy jak pracować "zdalnie" z Unixem/Linuxem siedząc przy maszynie z MS Windows
  - z programami "tekstowymi" nie ma problemu: wystarczy się wtelnetować do maszyny z linux-em ...
  - aby używać programów linuxowych/Xwindowsowych trzeba zainstalować na lokalnej maszynie XSerwer !
    np. program WinaXe v6.0
  - następnie telnetujemy się do linuxa; do zmiennej DISPLAY wpisujemy adres IP naszej lokalnej maszyny (odczytamy go poleceniem ipconfig)

    ```
    export DISPLAY=lokalny_adr_IP:0.0
    ```
  - i możemy włączać programy linuxowe/Xwindowsowe, które będą otwierały swoje okna na naszej lokalnej maszynie (będziemy widzieć zarówno okna zwykłych lokalnie uruchomionych programów Windows ORAZ okna programów uruchomionych na linuxie ...)
  - następny slajd:
    równoczesne używanie MS Word-a i programu OOWriter z Linuxa !!!

Zdalna praca z Unix-em