Minix3

Jan Świeżyński

9 grudnia 2008

Idee przyświecające projektowi

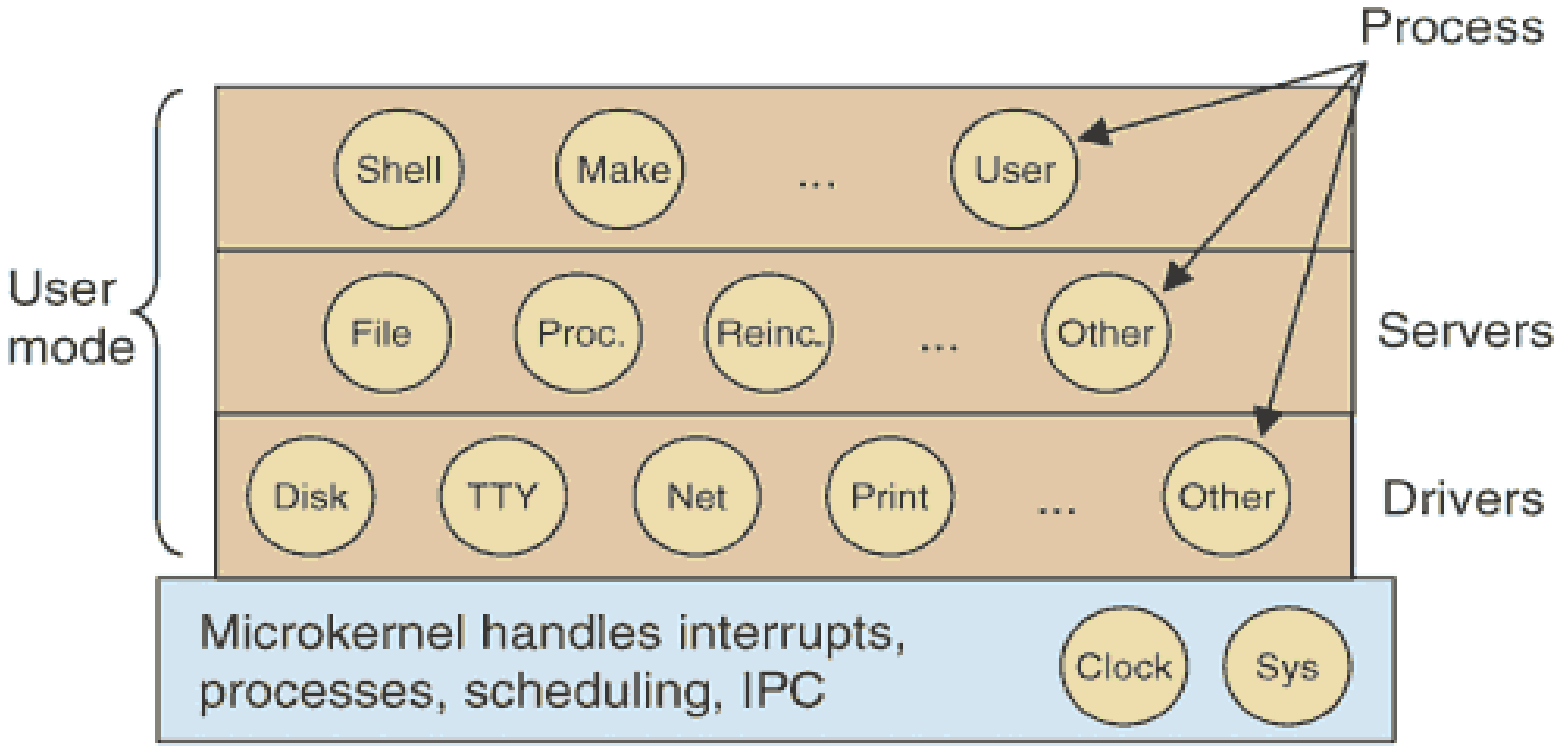
- Zwiększona ilość „zwykłych” użytkowników komputera, którzy oczekują od systemu niezawodności.
- Stworzenie systemu, którego jądro byłoby niewielkie (5000 linii kodu, przy 5 mln jądra Windowsa XP i 2,5 mln jądra Linuxa).
- Dodanie właściwości self-repairing.

Zarys ogólny

- Małe, zaufane mikrojądro.
- Serwery i sterowniki w przestrzeni użytkownika.

Zarys ogólny

- Małe, zaufane mikrojądro.
- Serwery i sterowniki w przestrzeni użytkownika.



Procesy i komunikacja międzyprocesorowa

- Procesy podobne do standardowych UNIX-owych.
- Komunikacja synchroniczna na zasadzie spotkań.
- Dodatkowy mechanizm powiadomień.

Przestrzeń jądra

- Odpowiedzialność mikrojądra - podstawowe mechanizmy, spośród tych, za które odpowiada jądro monolityczne.
- Sterownik zegara - niezbędny do kolejkowania procesów.
- System task - obsługa wywołań kierowanych do jądra.

Przestrzeń użytkownika

- Warstwa sterowników, w której leżą wszystkie sterowniki poza sterownikiem zegara.
- Warstwa serwerów - poszczególne serwery zajmują się obsługą POSIX-owych wywołań.

Specjalne serwery

- Reincarnation server.
- Data store.

Reincarnation server

- Proces macierzysty wszystkich pozostałych procesów przestrzeni użytkownika.
- Żądanie zwrócenia statusu wysyłane do działających procesów.

Data store

- Serwer mini bazy danych.
- Odzyskiwanie utraconych stanów po restarcie komponentu.
- Informowanie zależnych procesów.

Działanie właściwości „self-repairing”

- Integracja sterowników, serwerów i aplikacji.

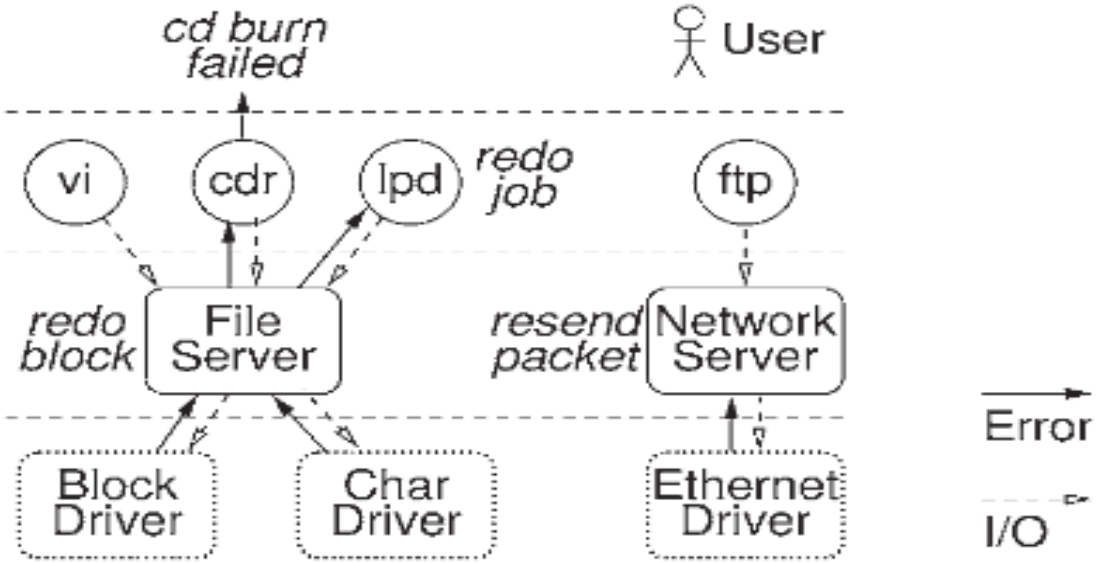


Figure 2: Recovery scenarios for device driver for block, character, and Ethernet devices. Errors are pushed up when they cannot be handled at the current level.

Wydajność systemu

- Nie najważniejsza - głównym celem zwiększenie niezawodności.

Wydajność systemu

- Nie najważniejsza - głównym celem zwiększenie niezawodności.

Call	Kernel	User	Δ	Ratio
getpid	0.831	1.011	0.180	1.22
lseek	0.721	0.797	0.076	1.11
open+close	3.048	3.315	0.267	1.09
read 64k+lseek	81.207	87.999	6.792	1.08
write 64k+lseek	80.165	86.832	6.667	1.08
creat+wr+del	12.465	13.465	1.000	1.08
fork	10.499	12.399	1.900	1.18
fork+exec	38.832	43.365	4.533	1.12
mkdir+rmdir	13.357	14.265	0.908	1.07
rename	5.852	6.812	0.960	1.16
Average				**1.12**

Figure 5: System call times for kernel-mode versus user-mode drivers. All times are in microseconds.

Program	2.0.4	3.0.1	Δ	Ratio
Build image	3.630	3.878	0.248	1.07
Build POSIX tests	1.455	1.577	0.122	1.08
Sort	99.2	103.4	4.2	1.04
Sed	17.7	18.8	1.1	1.06
Grep	13.7	13.9	0.2	1.01
Prep	145.6	159.3	13.7	1.09
Uuencode	19.6	21.2	1.6	1.08
Average				**1.06**

Figure 8: Run times in seconds for various test programs. The first two test were repeatedly run a loop, while the others were run only once to exclude effects from the file system's cache.