Klasyczny Mac OS

System, w którym użytkownik był administratorem pamięci, a każdy program mógł zabić każdy inny — i właśnie dlatego dało się go tak głęboko przerabiać.

Siedemnaście lat historii systemu, który nigdy nie miał ochrony pamięci, nigdy nie wywłaszczył żadnego procesu i przez ostatnie siedem lat życia był po części emulowany na własnym, natywnym sprzęcie. Opowieść o tym, jak jedna decyzja projektowa z 1983 roku — tablica pułapek trzymana w pamięci operacyjnej — dała jednocześnie najbardziej podatny na przeróbki system osobisty w historii i najbardziej nieznośną kategorię awarii, jaką wymyślono. O folderach, które były atrybutem pliku, a nie węzłem drzewa. O tym, dlaczego numer wersji bywa instrumentem prawnym. I o czterech następcach, z których żaden nie wyszedł poza fazę planów, slajdów albo kompilacji, która sypała się sama z siebie.

> Umiejscowienie w serii: to jest strona o **klasycznym Mac OS** — o samym systemie operacyjnym, od Systemu 1 z 1984 roku do Mac OS 9.2.2 z grudnia 2001. Xerox PARC, Alto, Smalltalk, Lisa i cała historia powstania projektu Macintosh są w tekście **Od Xeroxa do macOS** i nie powtarzam ich tutaj. System, który klasycznego Maca zastąpił, opisują **NeXTSTEP i OPENSTEP** oraz **Mac OS X: przejście**; stan dzisiejszy — **macOS dzisiaj**. Równoległy świat pecetowy tych samych lat jest w tekstach **Windows 95** i **OS/2**, a rodowód warstwy uniksowej, która ostatecznie wygrała u Apple — w tekście **BSD**.

Skróty użyte na tej stronie (43)

Skrót	Co znaczy
ADB	Apple Desktop Bus: magistrala klawiatury i myszy Maca, rozpoznawana bez konfiguracji
AFP	Apple Filing Protocol: protokół udostępniania plików Apple, rozumiejący oba widelce
API	Application Programming Interface: zestaw wywołań, z których korzysta program
ASCII	American Standard Code for Information Interchange: kodowanie, według którego sortowano nazwy rozszerzeń
AV	Audio-Video: oznaczenie Maców z rozbudowanym dźwiękiem i wideo, np. Quadra 840AV
BFS	Be File System: własny system plików BeOS-a
BSD	Berkeley Software Distribution: uniksowa linia, której warstwa ostatecznie wygrała u Apple
CD	Compact Disc: płyta optyczna; Mac OS 9.1 wypalał ją w Finderze
CMI	Computer Musical Instrument: syntezator-sampler Fairlight, na którym powstał dźwięk startowy Maców 5200–6300

Skrót	Co znaczy
DOS	Disk Operating System: tekstowy system pecetów
DR DOS	Digital Research DOS: system Novella, na którym miał działać Star Trek
DTP	Desktop Publishing: komputerowy skład publikacji, dziedzina, w której Mac królował
FAT	File Allocation Table: system plików DOS-a i Windows, bez miejsca na widelec zasobów
FTP	File Transfer Protocol: protokół przesyłania plików, który gubił widelec zasobów
GCR	Group Coded Recording: sposób zapisu na dyskietkach Maca 400 i 800 KB
HFS, HFS+	Hierarchical File System: system plików z B-drzewem z 1985 roku; HFS+ z 32-bitowymi blokami od 1998
HP	Hewlett-Packard: producent komputerów, udziałowiec Taligenta od 1994 roku
IBM	International Business Machines: partner Apple w sojuszu PowerPC i w Taligencie
IRQ	Interrupt Request: numer przerwania, który w pecetach ustawiało się ręcznie
ISO	International Organization for Standardization: autor normy ISO Latin-2 dla polskich znaków
JDK	Java Development Kit: pakiet Javy, w którym obsługę Unicode przygotowali ludzie z Taligenta
LC	Low-cost Color: tania, kolorowa linia Macintoshy
LQ	Letter Quality: jakość „listowa”, w nazwie drukarki ImageWriter LQ
MFS	Macintosh File System: płaski system plików Systemu 1, z folderami tylko na ekranie
MMU	Memory Management Unit: jednostka zarządzania pamięcią, używana przez pamięć wirtualną Systemu 7
MP	Multiprocessing: wieloprocesorowość, jak w zadaniach MP i w DayStar Genesis MP
NFS	Network File System: uniksowy protokół plików sieciowych, nieznający widelców
NTFS	New Technology File System: system plików Windows z alternatywnymi strumieniami
OS	Operating System: system operacyjny, jak w nazwie Mac OS
OS/2	Operating System/2: system IBM, równoległy świat pecetowy tych lat
PARC	Palo Alto Research Center: laboratorium Xeroxa, od którego zaczyna się historia Maca
PC	Personal Computer: pecet zgodny z IBM PC
PCI	Peripheral Component Interconnect: magistrala rozszerzeń w Macach od 1995 roku
PEF	Preferred Executable Format: format kodu PowerPC w klasycznym Mac OS
QEMU	Quick Emulator: wolny emulator, który uruchamia Mac OS 9.2 na emulowanym PowerMacu
ROM	Read-Only Memory: pamięć stała na płycie głównej, w której siedział Toolbox
SCSI	Small Computer System Interface: magistrala dysków od Macintosha Plus
SE	System Expansion: Macintosh SE z 1987 roku, kompakt z gniazdem rozszerzeń

Skrót	Co znaczy
SMB	Server Message Block: protokół udostępniania plików Windows
SOM	System Object Model: obiektowy model komponentów IBM, jedna z technologii Coplanda
USA, USD	Stany Zjednoczone i dolar amerykański
VIA	Versatile Interface Adapter: układ MOS 6522, który w Macu 128K generował dźwięk startu
WWDC	Worldwide Developers Conference: doroczna konferencja Apple dla programistów

Leksykon: nazwy i pojęcia z tej strony (48)

Nazwa	Co to jest
A-line (pułapki)	Słowa zaczynające się od bitów 1010, przez które program wołał Toolbox: procesor zgłaszał wyjątek, a Trap Dispatcher skakał pod adres z tablicy pułapek.
Apple Events	Mechanizm komunikacji między aplikacjami z Systemu 7, na którym stanął AppleScript.
AppleDouble	Zapis widelca zasobów w osobnym, ukrytym pliku z przedrostkiem ._.
AppleScript	Język skryptowy Apple z 1993 roku (System 7.1.1), zbudowany na Apple Events.
BeOS	Wywłaszczający, wielowątkowy system Be Inc., rozważany przez Apple w 1996 roku, ale niekompletny.
BinHex, MacBinary	Formaty pakujące oba widelce pliku w jeden strumień; stąd pliki .hqx i .bin.
Blue Meanies	Zespół, który zbudował System 7 z pomysłów zapisanych na niebieskich fiszkach.
Bomba systemowa	Okno awarii z ikoną bomby, numerem błędu (wyjątkiem procesora) i przyciskiem Restart.
Carbon	Podzbiór API klasycznego Toolboxa, działający w Mac OS 9 i natywnie w Mac OS X.
Classic	Warstwa zgodności Mac OS X uruchamiająca cały Mac OS 9 jako jeden proces; tylko na PowerPC, do 10.4.
Conflict Catcher	Narzędzie Casady & Greene, które metodą połowienia wskazywało rozszerzenie winne konfliktu.
Copland	Następca klasycznego Mac OS z ochroną pamięci i wywłaszczaniem, rozwijany w latach 1994–1996 i zamrożony.
Emulator 68k	Wbudowany w system dla PowerPC emulator starego kodu; wersja Erica Trauta rekompilowała go dynamicznie.
Extensions Manager	Panel z System 7.5 do włączania i wyłączania rozszerzeń oraz zapisywania ich zestawów.
Gershwin	Planowany następca Coplanda z wywłaszczaniem dla wszystkich aplikacji; nie wyszedł poza slajdy.
Get Info (Informacje)	Okno Findera, w którym użytkownik ręcznie ustawiał przydział pamięci aplikacji.

Nazwa	Co to jest
INIT (rozszerzenie)	Plik z zasobem INIT, ładowany przy starcie w kolejności alfabetycznej i łatający wywołania systemowe.
Konflikt rozszerzeń	Awaria wynikająca z tego, że dwa rozszerzenia łatają tę samą pułapkę i przeszkadzają sobie nawzajem.
Mac OS Central European	Kodowanie polskich i innych środkowoeuropejskich znaków w Mac OS, obsługiwane przez WorldScript.
Makówka	Potoczna polska nazwa Macintosha w latach dziewięćdziesiątych.
Marklar	Projekt Mac OS X na procesorach Intela, ujawniony publicznie w 2005 roku.
Mazovia	Polska strona kodowa z pecetów, jedno z czterech kodowań, między którymi konwertowano teksty.
Mini vMac, Basilisk II, SheepShaver	Emulatory: wczesnego Maca z Systemem 6, Maców 68k oraz Maców PowerPC do Mac OS 9.0.4.
Mixed Mode Manager	Warstwa przełączająca wykonanie między kodem 68k a PowerPC przy każdym przekroczeniu granicy.
MultiFinder	Dodatek z 1987 roku (System Software 5.0) trzymający kilka aplikacji naraz; od Systemu 7 na stałe.
Multiprocessing Services	Wywłaszczające zadania od 1996 roku, które nie mogły wołać Toolboxa, więc nadawały się tylko do liczenia.
Nanokernel	Cienka warstwa pod klasycznym systemem na PowerPC; w Mac OS 8.6 z wywłaszczającym planistą.
Pink	Projekt obiektowego systemu z różowych fiszek z 1988 roku, z którego w 1992 roku powstał Taligent.
Platinum	Szary, trójwymiarowy wygląd interfejsu z Mac OS 8, zaprojektowany pierwotnie dla Coplanda.
Power Computing	Producent klonów Maca, szybszych i tańszych od oryginałów; Apple wykupiło go w 1997 roku.
QuickDraw GX	Rozbudowana biblioteka graficzna Apple, jedna z technologii dokładanych do Coplanda.
ResEdit	Darmowy edytor zasobów Apple, którym nieprogramiści zmieniali i tłumaczyli cudze programy.
Sad Mac	Ikona smutnego Maca z kodem błędu, pokazywana, gdy test sprzętu przy starcie się nie powiódł.
Sherlock	Wyszukiwarka z Mac OS 8.5 łącząca przeszukiwanie dysku, treści plików i Internetu.
SIZE (zasób)	Zasób aplikacji z preferowanym i minimalnym przydziałem pamięci oraz flagami zdolności.
Star Trek	Tajny projekt Apple i Novella z lat 1992–1993: System 7 na komputerach z procesorami x86.
System Enablers	Małe rozszerzenia dodające obsługę nowego modelu Maca bez nowej wersji systemu.
Taligent	Spółka Apple i IBM z 1992 roku, rozwiązana w 1997; przetrwała z niej m.in. biblioteka ICU.

Nazwa	Co to jest
Toolbox	Biblioteka interfejsu i grafiki zapisana w 64 KB ROM pierwszego Maca, łatana przez plik System.
TrueType	Skalowalne czcionki konturowe wbudowane w System 7.
Tłuste binaria (fat binaries)	Plik aplikacji z kompletnym kodem 68k i PowerPC naraz, mniej więcej dwa razy większy.
Uchwyt (handle)	Wskaźnik na wskaźnik główny, pozwalający systemowi przesuwać bloki relokowalne; blokowany przez HLock.
WaitNextEvent	Funkcja Menedżera Zdarzeń, przy której wywołaniu aplikacja oddawała procesor innym.
Widelec zasobów	Druga, ustrukturyzowana część pliku z typowanymi zasobami, obok widelca danych.
Wielozadaniowość kooperacyjna	System nie odbiera procesora; aplikacje muszą oddawać go same.
WorldScript	Warstwa System 7.1 obsługująca systemy pisma inne niż rzymski, w tym kodowanie środkowoeuropejskie.
Wymiana dysków (disk swapping)	Wkładanie na zmianę dyskietki źródłowej i docelowej przy kopiowaniu na Macu z jedną stacją.
Świat A5	Obszar partycji aplikacji adresowany rejestrem A5: zmienne globalne i tablica skoków.

> **Zastrzeżenie zakresu.** To nie jest katalog modeli Macintosha ani historia firmy Apple. Sprzęt pojawia się wyłącznie tam, gdzie tłumaczy decyzje systemowe — 64 kilobajty ROM-u, brak dysku twardego, przejście na PowerPC. Przejęcie NeXT opisuję tylko jako domknięcie tej historii: mechanika samego NeXTSTEP-u i budowa Mac OS X są na osobnych stronach. Numery wersji podaję tak, jak je podaje źródło, i osobno tłumaczę, dlaczego tabele wersji z tamtych lat bywają niespójne.

Wprowadzenie: jedno źródło zalet i wad

Klasyczny Mac OS jest w tej serii przypadkiem szczególnym, bo daje się go opisać jedną tezą, z której wynika dosłownie wszystko inne — i dobre, i złe. Teza brzmi tak: **był to system, w którym użytkownik pełnił funkcję administratora pamięci, a każdy działający program miał techniczną możliwość zabicia każdego innego programu i samego systemu.** Nie było ochrony pamięci, nie było wywłaszczania, nie było granicy uprawnień, nie było rejestru tego, kto co modyfikuje. Wszystko działało w jednej przestrzeni adresowej i wszystko było dla wszystkiego dostępne.

Pierwszy odruch człowieka wychowanego na systemach z izolacją procesów jest taki, że to opis wady. To jest opis wady. Ale to jest dokładnie ten sam opis, z którego wynikają zalety, i to jest cała pointa tego tekstu. Skoro tablica funkcji systemowych leżała w pamięci operacyjnej i dowolny kod mógł podmienić w niej adres, to dowolny kod mógł też rozszerzyć system o funkcję, której Apple nie przewidziało — i robił to bez źródeł, bez rekompilacji systemu, bez zgody producenta. Skoro wszystkie napisy interfejsu programu leżały w ustrukturyzowanym widelcu zasobów, to dowolny użytkownik z darmowym narzędziem Apple mógł ten program przetłumaczyć na polski, nie mając kodu źródłowego. Skoro pamięć przydzielało się aplikacji ręcznie, to użytkownik mógł naprawić cudzy błąd programistyczny, po prostu dokładając megabajt.

Te dwie listy — lista możliwości i lista katastrof — mają wspólne źródło i nie dało się ich rozdzielić. Konflikt rozszerzeń nie był awarią systemu łatania pułapek; był tym systemem, użytym przez dwie osoby naraz. Komunikat „za mało pamięci” przy czterdziestu wolnych megabajtach nie był błędem modelu partycji; był tym modelem, zobaczonym przez użytkownika, który nie wiedział, że jest administratorem pamięci.

> Jeśli ta strona ma jedno zdanie do zapamiętania, to takie: **wszystkie zalety i wszystkie wady klasycznego Mac OS wynikają z tej samej decyzji — że nie ma granicy między kodem systemu a kodem cudzym.** Każdy kolejny rozdział tego tekstu jest rozwinięciem jednej z dwóch stron tej decyzji.

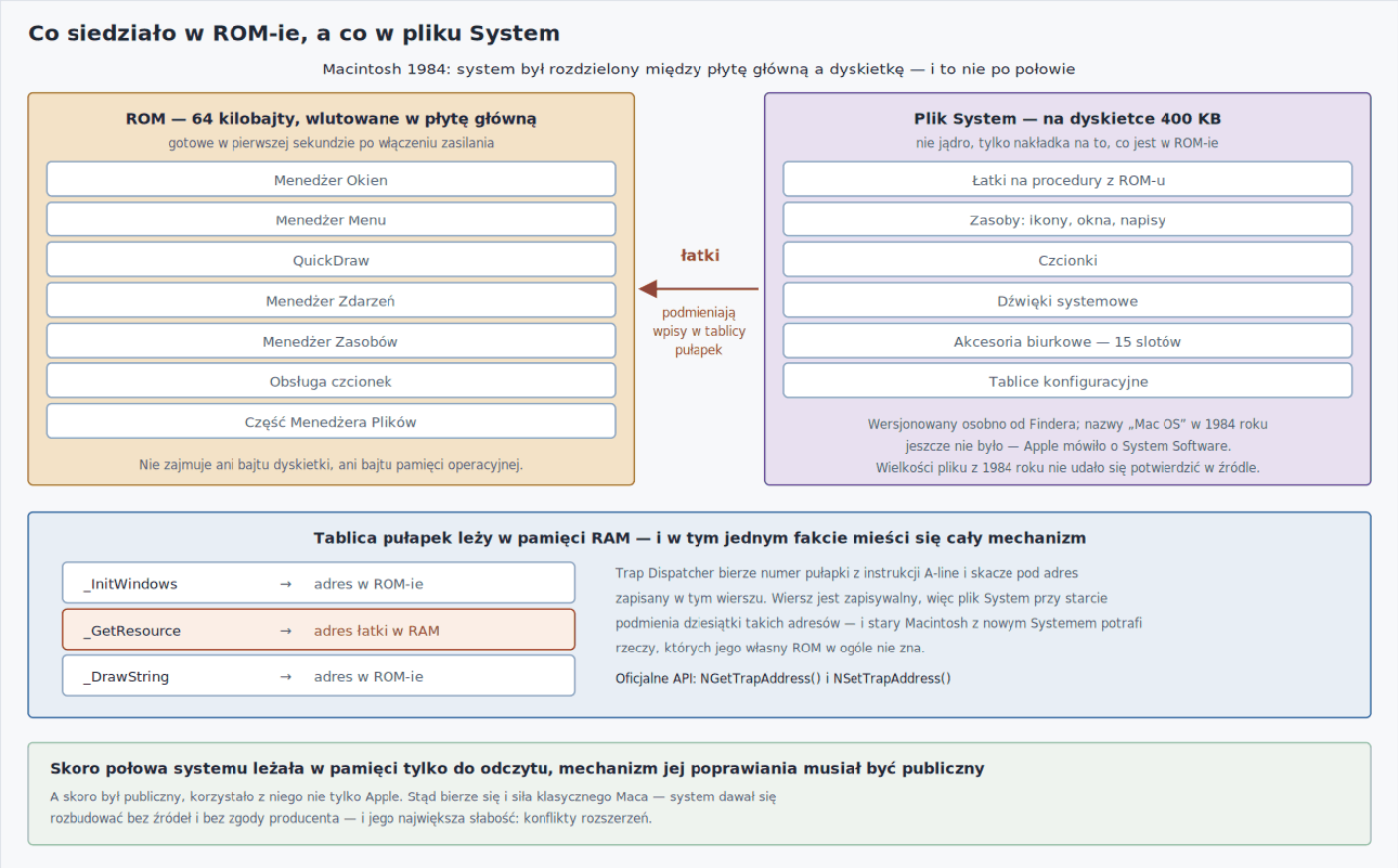
System 1 i pytanie źle postawione

Najczęstsze pytanie o pierwszego Macintosha brzmi: jak to się wszystko mieściło w czterystu kilobajtach dyskietki? Odpowiedź, którą trzeba dać na początku, żeby nie kłamać przez resztę tekstu, brzmi: **nie mieściło się**. Pytanie jest źle postawione, bo zakłada, że na dyskietce startowej był system operacyjny. Nie było go tam. Na dyskietce była jego mniejsza połowa.

Pierwszy Macintosh, ten ze stycznia 1984 roku, miał 128 kilobajtów pamięci operacyjnej, procesor Motorola 68000 taktowany zegarem 7,8336 MHz, wbudowany ekran o rozdzielczości 512×342 piksele — czarno-biały, bez odcieni szarości — i jedną wbudowaną stację dyskietek 3,5 cala o pojemności 400 kilobajtów, jednostronną, z zapisem GCR i zmienną prędkością obrotową. Nie było dysku twardego. Nie było drugiej stacji w standardzie. Nie było slotów rozszerzeń. Na dyskietce startowej musiał się zmieścić plik `System`, plik `Finder`, sterownik drukarki `ImageWriter`, zasoby czcionek i akcesoria biurkowe — a użytkownik chciał tam jeszcze zmieścić aplikację i własne dokumenty.

Rozwiązaniem było przeniesienie połowy systemu poza dyskietkę — i poza pamięć operacyjną. Macintosh miał **64 kilobajty pamięci tylko do odczytu** wlutowanej w płytę główną, a w niej zaszyty kompletny **Toolbox**: menedżer okien, menedżer menu, QuickDraw, menedżer zdarzeń, menedżer zasobów, obsługę czcionek i część menedżera plików. To wszystko było na płycie, gotowe do wywołania w pierwszej sekundzie po włączeniu zasilania, i nie zajmowało ani bajtu dyskietki ani — co ważniejsze przy 128 kilobajtach — ani bajtu pamięci operacyjnej.

Plik `System` nie był więc jądrem. Był **nakładką**: zbiorem zasobów, łatek na procedury zaszyte w ROM-ie, czcionek, dźwięków, akcesoriów biurkowych i tablic konfiguracyjnych. Ta różnica nie jest niuansem terminologicznym, bo z niej wynikają trzy rzeczy, które przewijają się przez cały ten tekst. Po pierwsze: skoro system był nakładką na ROM, to stary Macintosh z nowym plikiem `System` potrafił rzeczy, których jego własny ROM nie znał — Apple po prostu dokładało łatki. Po drugie: skoro połowa systemu była w pamięci tylko do odczytu, to tej połowy nie dało się poprawić inaczej niż właśnie przez łatanie, a mechanizm łatania musiał być oficjalny, dostępny i udokumentowany. Po trzecie: skoro ten mechanizm był oficjalny i dostępny, to korzystali z niego wszyscy, nie tylko Apple. O tym jest cała sekcja o pułapkach systemowych.



Rysunek 1. *Rozkład sił, który tłumaczy całą resztę: kod systemowy siedział w pamięci stałej na płycie głównej, a plik na dyskietce służył głównie do tego, żeby ten kod poprawiać — i dlatego mechanizm poprawiania musiał być publiczny.*

> **Uwaga na liczby.** Po forach retro krążą w kilku wariantach twierdzenia w rodzaju „System 1.0 zajmował 216 kilobajtów". Nie udało mi się potwierdzić żadnego z nich w wiarygodnym źródle i dlatego nie podaję tu żadnej konkretnej wielkości pliku `System` z 1984 roku. Kto chce liczbę pewną, powinien zważyć realny obraz dyskietki w emulatorze — to zajmuje kwadrans i daje wartość zmierzoną zamiast przepisanej.

Nazwa, której wtedy nie było

W 1984 roku nie istniała nazwa „Mac OS”. Apple mówiło o **System Software**, a wersje numerowało osobno dla pliku `System` i osobno dla pliku `Finder`. Pierwsze wydanie to **System 1.0 / Finder 1.0** ze stycznia 1984. Nazwa „System 1” jest retronimem — powstała później, kiedy trzeba było jakoś uporządkować historię. Podobnie „klasyczny Mac OS” to określenie ukute dopiero po premierze Mac OS X, żeby odróżnić dwie zupełnie różne rzeczy noszące podobną nazwę.

MFS: foldery jako atrybut pliku, nie węzeł drzewa

To jest fragment, który warto opisać dokładnie, bo jest jednocześnie bardzo ciekawy i prawie zawsze opisywany źle.

System 1 używał systemu plików **MFS** — Macintosh File System — i ten system plików był **płaski**. Wszystkie informacje o plikach i katalogach na woluminie trzymane były w jednym pliku katalogowym. Nie istniała struktura drzewiasta: wszystkie pliki na dyskietce leżały logicznie obok siebie, w jednym worku, a przeszukiwanie polegało na przejściu całej listy.

Użytkownik mimo to **widział foldery** i mógł ich używać. Każdy plik miał w swoim rekordzie katalogowym pole wskazujące, do którego folderu należy, a Finder rysował na tej podstawie hierarchię, która na ekranie wyglądała dokładnie jak drzewo. Nie była drzewem. Folder był **atrybutem pliku** — takim samym w swojej naturze jak data modyfikacji albo typ — a nie węzłem struktury.

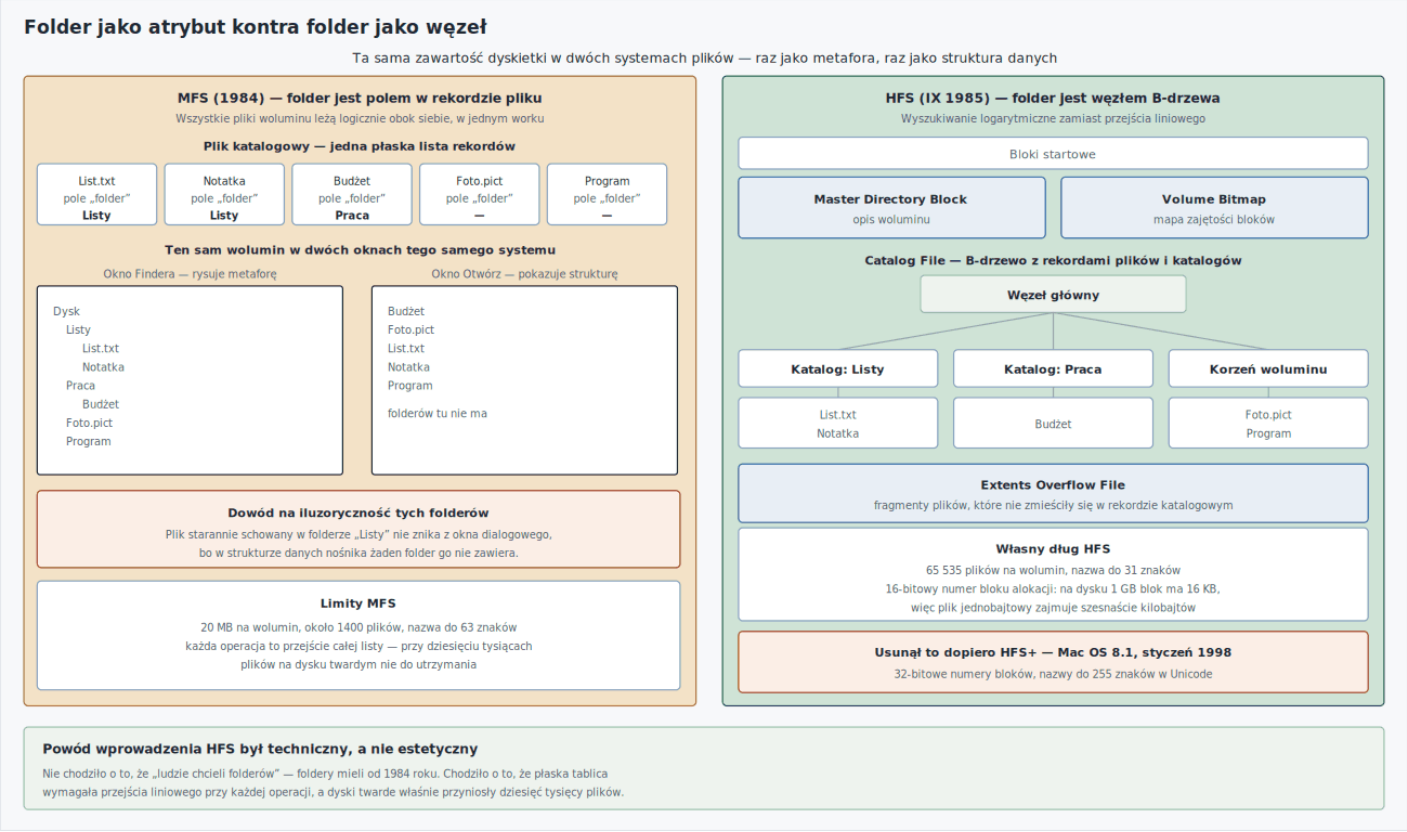
Najlepszy dowód na iluzoryczność tych folderów jest zarazem najładniejszym szczegółem w całej historii wczesnego Maca: **foldery pojawiały się w oknach Findera, ale nie pojawiały się w systemowych oknach dialogowych Otwórz i Zapisz**. Standardowy dialog plikowy pokazywał płaską listę wszystkich plików woluminu, bo taki był rzeczywisty stan rzeczy na nośniku. Użytkownik mógł więc starannie schować plik do folderu „Listy” i nadal widzieć go w oknie Otwórz obok wszystkich pozostałych, w jednej długiej liście. Dwa okna tego samego systemu pokazywały dwa różne modele świata, bo jedno z nich rysowało metaforę, a drugie strukturę danych.

Mechanizm tworzenia folderów był równie charakterystyczny. **Nie było polecenia „Nowy folder”.** Na każdym woluminie istniał zawsze jeden pusty folder o nazwie `Empty Folder`. Kiedy użytkownik go przemianował albo cokolwiek do niego wrzucił, Finder natychmiast tworzył obok kolejny pusty folder o tej samej nazwie. Był to zatem wieczny, samoodradzający się byt — coś pomiędzy szablonem a zjawiskiem przyrodniczym. Polecenie **New Folder** pojawiło się dopiero w Systemie 2.0 w kwietniu 1985 i zakończyło erę wiecznego pustego folderu.

Pulpit Systemu 1 z Finderem. Zwróć uwagę, że foldery w oknie wyglądają jak drzewo, choć w MFS by

Pulpit Systemu 1 z Finderem. Zwróć uwagę, że foldery w oknie wyglądają jak drzewo, choć w MFS były tylko polem w rekordzie pliku — i że cały interfejs mieści się na czarno-białym ekranie 512 × 342. Zrzut ekranu: Apple; przytoczony w celu omówienia (prawo cytatu, art. 29 pr. aut.). Plik: en.wikipedia, non-free.

Ograniczenia MFS wyglądały następująco: maksymalny rozmiar woluminu **20 megabajtów**, maksymalnie około **1400 plików** na woluminie, nazwa pliku do 255 znaków w strukturze danych (Finder ograniczał to praktycznie do 63, a potem do 31 znaków) i dwukropek zakazany jako znak, bo pełnił funkcję separatora ścieżki. Przy dyskietce czterystukilobajtowej limit dwudziestu megabajtów wyglądał absurdalnie wysoko — dokładnie ten typ decyzji projektowej, który po dwóch latach staje się problemem.



Rysunek 2. *Ta sama zawartość dyskietki widziana dwa razy: w MFS hierarchia istnieje wyłącznie jako pole w rekordzie pliku i dlatego rozpada się wszędzie tam, gdzie nie rysuje jej Finder; w HFS jest realną strukturą danych, którą widzą wszystkie warstwy systemu.*

Jedno zadanie naraz i akcesoria biurkowe

System 1 uruchamiał jedną aplikację naraz. Zakończenie pracy aplikacji oznaczało powrót do Findera i był to powrót dosłowny: Finder ładowano z dysku od nowa. Nie było przełączania między programami, nie było menu aplikacji, nie było pojęcia tła. Model pracy wyglądał tak, że włącza się jedną rzecz, robi się w niej to, co się ma do zrobienia, wychodzi się z niej i włącza następną.

Obejściem były **akcesoria biurkowe** — Desk Accessories, w skrócie DA. Kalkulator, Notatnik, Zegar, Wybieracz: małe moduły kodu ładowane do przestrzeni adresowej bieżącej aplikacji i wywoływane z menu Apple. Nie były osobnymi procesami. Dzieliły pamięć i procesor z gospodarzem, czyli z programem, w którym akurat pracował użytkownik. Limit był twardy — klasycznie piętnaście slotów w pliku `System` — a instalowało się je osobnym narzędziem o nazwie **Font/DA Mover**.

I tu pojawia się rzecz, która dla człowieka przyzwyczajonego do katalogu z czcionkami brzmi niedorzecznie: czcionki i akcesoria biurkowe **nie były plikami**. Były zasobami wewnątrz pliku `System`. Dodanie czcionki do systemu oznaczało przeniesienie zasobu z jednego pliku do drugiego i wymagało dedykowanego programu, który umiał to zrobić bez uszkodzenia niczego po drodze. Dopiero System 7 zrobił z czcionek zwykłe pliki, które przeciąga się do folderu — i była to jedna z tych zmian, które w zestawieniu funkcji wyglądają nudno, a w codziennej pracy znaczyły więcej niż połowa reszty.

Tryb wymiany dysków, czyli udręka o własnej nazwie

Przy jednej stacji dyskietek i 128 kilobajtach pamięci operacyjnej skopiowanie pliku z dyskietki A na dyskietkę B musiało iść przez pamięć. Bufor był mały, więc system prosił o dyskietkę źródłową, czytał kawałek, prosił o docelową, zapisywał, prosił znów o źródłową — i tak do skutku. Przy pełnej dyskietce oznaczało to kilkadziesiąt wymian nośnika, z których każda wymagała fizycznego wyjęcia jednej dyskietki i włożenia drugiej.

Zjawisko zyskało w folklorze macowym nazwę **disk swapping**, bywało też nazywane „macintoshowym tańcem”. Ikona dyskietki z nazwą woluminu, wyświetlana na środku ekranu z prośbą o włożenie konkretnego nośnika, jest jednym z najbardziej rozpoznawalnych obrazów wczesnego Maca — i jednym z niewielu elementów tamtego interfejsu, które wspomina się bez czułości.

Macintosh 512K z września 1984, nazywany „Fat Mac”, drastycznie zmniejszył liczbę wymian, bo pamięci było cztery razy więcej. Zewnętrzna stacja dyskietek była w tamtych latach jednym z najczęściej kupowanych akcesoriów — i kupowano ją nie dla pojemności, tylko dla świętego spokoju.

> **Anegdota, której nie potwierdzam liczbowo.** Powszechnie powtarza się, że skopiowanie pełnej dyskietki na Macintoshu 128K wymagało pięćdziesięciu do sześćdziesięciu wymian. Rachunek to uprawdopodabnia — czterysta kilobajtów podzielone przez rząd ośmiu kilobajtów użytecznego bufora daje mniej więcej tyle — ale nie znalazłem twardego źródła Apple z tą liczbą. Traktuję ją jako szacunek zgodny z arytmetyką, nie jako ustalenie.

Wydanie po wydaniu

Przed opisem architektury warto mieć przed oczami cały ciąg wydań, bo inaczej nie widać, w którym momencie która decyzja zaczęła uwierać. Tabela poniżej podaje te wydania, które coś realnie zmieniły; pomijam drobne poprawki i wersje dedykowane pojedynczym modelom. Daty dzienne podaję tam, gdzie są pewne.

Wydanie	Data	Co wniosło
System 1.0 / Finder 1.0	I 1984	MFS, jedno zadanie, akcesoria biurkowe, Toolbox w 64 KB ROM
System 1.1	V 1984	Poprawki, szybszy start, okno „About the Finder”
System 2.0 / Finder 4.1	IV 1985	Polecenia New Folder i Shut Down, LaserWriter, AppleTalk, MiniFinder
System 2.1	IX 1985	HFS, ale wyłącznie dla dysku Hard Disk 20; dyskietki nadal MFS
System 3.0 / Finder 5.1	I 1986	HFS wbudowany w ROM, dyskietki 800 KB, SCSI, Kosz zachowujący zawartość
System 4.0 / Finder 6.0	I 1987	Macintosh SE i II, wiele monitorów, początki koloru
System 4.1 / Finder 5.5	IV 1987	Obsługa dysków większych niż 32 MB
System Software 5.0	X 1987	MultiFinder; w środku System 4.2 i Finder 6.0
System 6.0	10 VI 1988	Zintegrowany MultiFinder, QuickerGraf, MacroMaker, pełniejszy kolor
System 6.0.8	13 V 1991	Ostatnie wydanie linii, tego samego dnia co System 7
System 7.0	13 V 1991	32 bity, pamięć wirtualna, aliasy, Apple Events, TrueType, File Sharing
System 7.1	VIII 1992	Folder Fonts, WorldScript, System Enablers; pierwszy płatny osobno
System 7.1.1 („System 7 Pro”)	X 1993	AppleScript, PowerTalk, QuickTime w pakiecie
System 7.1.2	III 1994	Pierwsza wersja na PowerPC, fabrycznie z Power Macintoshami
System 7.5	IX 1994	Apple Guide, Launcher, WindowShade, Control Strip, Stickies, QuickDraw GX, PowerTalk
System 7.5.1	III 1995	Pierwszy ekran startowy z logo „Mac OS”
System 7.5.2	1995	Maszyny PCI, Open Transport zamiast MacTCP
System 7.5.3	1996	Duża aktualizacja zbiorcza; przez lata udostępniana przez Apple za darmo
System 7.5.5	IX 1996	Najstabilniejszy punkt linii 7.5; ostatnia wersja na procesory 68000

Wydanie	Data	Co wniosło
Mac OS 7.6	I 1997	Pierwsza wersja oficjalnie nazwana „Mac OS”; wymaga 68030
Mac OS 7.6.1	IV 1997	Ostatnie wydanie linii Systemu 7; koniec zapisu do MFS
Mac OS 8 („Tempo”)	26 VII 1997	Platinum, wielowątkowy Finder natywny dla PowerPC, menu kontekstowe; odcięcie klonów
Mac OS 8.1	I 1998	HFS+; ostatnia wersja działająca na 68040
Mac OS 8.5	X 1998	Pierwsza wersja wyłącznie dla PowerPC; Sherlock, motywy, Application Switcher
Mac OS 8.6	V 1999	Rozbudowany nanokernel, Multiprocessing Services 2.x
Mac OS 9	23 X 1999	Multiple Users, Keychain, Software Update, Sherlock 2, Carbon
Mac OS 9.1	I 2001	Wypalanie CD w Finderze; pierwsza wersja dobrze pracująca jako Classic
Mac OS 9.2.2	5 XII 2001	Ostatnia wersja klasycznego Mac OS w historii

System 2 i 3: hierarchia wchodzi naprawdę

System 2.0 z kwietnia 1985 przyniósł dwa polecenia, których brak dzisiaj brzmi jak żart: **New Folder** i **Shut Down**. Do tego doszła obsługa **LaserWritera** i sieci **AppleTalk** oraz możliwość startu z dysku twardego przez pośrednika zwanego MiniFinder. Rok 1985 jest w tej historii cezurą, bo LaserWriter plus AppleTalk plus Macintosh to jest dokładnie ten zestaw, z którego wyrosło DTP — i to on decyduje o polskim wątku pod koniec tego tekstu.

Najważniejsza zmiana tej gałęzi przyszła we wrześniu 1985 razem z dyskiem **Hard Disk 20**, podpinanym — co dziś wygląda ekscentrycznie — do portu stacji dyskietek. To wtedy po raz pierwszy pojawił się **HFS**, w formie kompromisowej: działał wyłącznie na tym jednym dysku, a kod ładowano z pliku leżącego na dyskietce. Dyskietki nadal były MFS. System 3.0 ze stycznia 1986, wydany razem z Macintoshem Plus, wbudował HFS **na stałe w ROM** i od tego momentu hierarchiczny system plików był uniwersalny. Przy okazji przyszły dyskietki 800 KB, magistrala SCSI i Kosz, który przestał sam się opróżniać.

Powód wprowadzenia HFS jest techniczny i warto go podać wprost, bo krąży wersja romantyczna („ludzie chcieli folderów”). Prawdziwy powód brzmi inaczej: płaska tablica katalogowa MFS

wymagała przejścia liniowego przy każdej operacji, a przy dziesięciu tysiącach plików na dysku twardym było to nie do utrzymania. HFS zastąpił ją **B-drzewem**, czyli wyszukiwaniem logarytmicznym. Wolumin HFS składa się z bloków startowych, **Master Directory Block** opisującego wolumin, **Volume Bitmap** z mapą zajętości bloków alokacji, **Catalog File** — właśnie tego B-drzewa z rekordami wszystkich plików i katalogów — oraz **Extents Overflow File** na fragmenty plików, które nie zmieściły się w rekordzie katalogowym.

HFS przyniósł też własny dług, i to taki, który uwierał do samego końca linii. Maksymalnie 65 535 plików na woluminie, nazwa pliku do 31 znaków — to są limity, z którymi dało się żyć. Ale najboleśniejsza konsekwencja projektu dotyczy czegoś innego: HFS używa **16-bitowego numeru bloku alokacji**, co oznacza, że każdy wolumin ma maksymalnie 65 535 bloków alokacji, niezależnie od swojej wielkości. Rozmiar bloku to rozmiar woluminu podzielony przez 65 535. Na dysku gigabajtowym blok alokacji ma **16 kilobajtów** — i plik o wielkości jednego bajta zajmuje na nim szesnaście kilobajtów.

> **To nie jest ciekawostka, tylko realna utrata pojemności.** Wolumin z dużą liczbą małych plików — a taki jest każdy wolumin systemowy i każdy wolumin z projektem DTP pełnym drobnych grafik — potrafił stracić na tym zjawisku znaczną część swojej pojemności. Standardową praktyką administracyjną było dzielenie dużych dysków na mniejsze partycje wyłącznie po to, żeby zmniejszyć blok alokacji. Problem rozwiązał dopiero HFS+ w Mac OS 8.1 ze stycznia 1998, wprowadzając 32-bitowe numery bloków alokacji.

System 4 i MultiFinder: wielozadaniowość wchodzi bokiem

System 4.0 ze stycznia 1987 przyszedł razem z Macintoshem SE i Macintoshem II, dołożył obsługę wielu monitorów i początki koloru; System 4.1 z kwietnia tego samego roku dodał obsługę dysków większych niż 32 megabajty. Rzecz istotna wydarzyła się jednak w październiku 1987 i nosiła nazwę **System Software 5.0**.

Zawartością tego pakietu był **MultiFinder** — i tu zaczyna się właściwa historia wielozadaniowości na Macintoshu. MultiFinder był **opcjonalny**: w oknie „Set Startup…” wybierało się między Finderem, który uruchamiał jedną aplikację i zostawiał jej całą wolną pamięć, a MultiFinderem, który trzymał kilka aplikacji naraz, każdą z mniejszym przydziałem. Na maszynie z jednym megabajtem pamięci MultiFinder bywał po prostu nieużyteczny, więc była to realna decyzja administracyjna, a nie kosmetyczny przełącznik.

MultiFinder wprowadził trzy rzeczy, które zostały do końca linii: **wielozadaniowość kooperacyjną**, pojęcie aplikacji pierwszoplanowej i tła oraz menu aplikacji w prawym górnym rogu ekranu. Wprowadził też czwartą, mniej widoczną, a bardzo ważną dla tego tekstu: **zasób** `SIZE` w pliku aplikacji. Zasób ten deklarował preferowany i minimalny przydział pamięci oraz flagi zdolności — `canBackground`, `acceptSuspendResumeEvents`, `onlyBackground`. To właśnie ten zasób odpowiadał za

wartości, które użytkownik widział potem w oknie „Informacje” i które mógł ręcznie zmienić.

System 6: stabilizacja i sufit

System 6.0 wydano **10 czerwca 1988**, a linia doczekała się ośmiu wydań poprawkowych — ostatnie, 6.0.8, wyszło **13 maja 1991**, tego samego dnia co System 7. Start był nieudany: Don Crabb określił w „BYTE” wersję 6.0 jako *buggy and unreliable*, a w wydaniu 6.0.1 naprawiono **66 błędów**. Mimo to System 6 zapracował sobie potem na reputację systemu „szybkiego i chudego” — na procesorze 68000 z jednym czy dwoma megabajtami pamięci działał żwawiej niż System 7 i część użytkowników została przy nim długo po 1991 roku.

System 6 dawał zintegrowany MultiFinder — koniec wyboru przy starcie — obsługę drukarek PostScript i ImageWriter LQ, **QuickerGraf** przyspieszający rysowanie, **MacroMaker** do nagrywania makr i pełniejszą obsługę koloru. Jego sufity były jednak niskie i wszystkie wynikały z wieku architektury: **8 megabajtów pamięci** przy adresowaniu 24-bitowym, **brak pamięci wirtualnej**, limit woluminu 2 gigabajty, okna dialogowe plików w zasadzie niezmienione od 1984 roku, brak aliasów i brak własnych ikon.

System 6.0.8, ostatnie wydanie linii, które wyszło tego samego dnia co System 7. Zwróć uwagę na p

System 6.0.8, ostatnie wydanie linii, które wyszło tego samego dnia co System 7. Zwróć uwagę na prostotę pulpitu — bez aliasów i bez własnych ikon, za to „szybki i chudy” na starszych maszynach. Zrzut ekranu: Apple; przytoczony w celu omówienia (prawo cytatu, art. 29 pr. aut.). Plik: en.wikipedia, non-free.

System 7 (1991): największy skok w historii linii

System 7.0 wydano **13 maja 1991** i jest to największa pojedyncza zmiana w całej historii klasycznego Mac OS. Nie „jedna z większych” — największa. Warto wyliczyć, co się w niej zmieściło, bo lista jest niebanalna.

32-bitowa przestrzeń adresowa. Limit pamięci rośnie z 8 megabajtów do teoretycznych 4 gigabajtów. Warunkiem było, by aplikacje i sterowniki były „32-bit clean”, czyli nie używały górnych ośmiu bitów adresu do własnych celów — praktyka legalna i rozpowszechniona w epoce adresowania 24-bitowego, a po 1991 roku będąca źródłem awarii.

MultiFinder przestaje być opcją. Wielozadaniowość kooperacyjna jest wbudowana na stałe i nie da się jej wyłączyć. Znika wybór „Finder czy MultiFinder”, a razem z nim znika możliwość oddania jednej aplikacji całej maszyny.

Apple Events. Systemowy mechanizm komunikacji międzyaplikacyjnej oparty na czterobajtowych kodach zdarzeń. To jest fundament, na którym stanął potem AppleScript — i tu trzeba uważać na datę, bo mit jest rozpowszechniony.

> **AppleScriptu nie było w Systemie 7.0.** System 7 wprowadził *fundament*, czyli Apple Events. Sam język pojawił się w 1993 roku, w pakiecie z System 7.1.1 („System 7 Pro”), a powszechnie dostępny stał się dopiero od System 7.5. Część opracowań — w tym skrótowe hasła encyklopedyczne — przypisuje AppleScript wprost do wydania z 1991 roku. To jest skrót myślowy, nie fakt.

TrueType. Skalowalne czcionki konturowe wbudowane w system, uzupełniające czcionki bitmapowe. Do tej pory ładny wydruk wymagał czcionek PostScript i programu **Adobe Type Manager**; TrueType dał wygładzone czcionki na ekranie i na drukarce bez dodatkowego oprogramowania. Dla rynku DTP miało to ogromne znaczenie, a dla relacji Apple z Adobe było aktem wojny.

Reorganizacja teczki systemowej. Zamiast wysypiska w korzeniu System Folder pojawiają się podfoldery: `Extensions`, `Control Panels`, `Preferences`, `Apple Menu Items`, `Startup Items`, a od 7.1 także `Fonts`. Menu Apple staje się konfigurowalne — wrzuca się do `Apple Menu Items` cokolwiek i to się tam pojawia. Kosz przestaje znikać przy restarcie, co brzmi banalnie, a zmieniło model mentalny usuwania plików: od tej pory „wyrzucone” znaczyło „leży w koszu”, a nie „zniknęło przy okazji”.

Koszt tego wszystkiego był realny. System 7 był wyraźnie cięższy od Systemu 6: minimum to 2 megabajty pamięci i 4 megabajty przestrzeni dyskowej, a komfort zaczynał się przy czterech do ośmiu megabajtach. Na procesorze 68000 — czyli na Macu Plus, SE i Classicu — działał, ale wolno. To jest właśnie ten moment, w którym System 6 zdobył swoją reputację.

System 7.1 i System Enablers

System 7.1 z sierpnia 1992 przyniósł folder `Fonts` — czcionki wreszcie jako pliki do przeciągnięcia — oraz **WorldScript**, warstwę obsługi systemów pisma innych niż rzymski. WorldScript jest istotny dla polskiego wątku i wrócę do niego na końcu tekstu. Była to też pierwsza wersja systemu, za którą Apple kazało sobie zapłacić jako za osobny produkt.

Drugą nowością były **System Enablers**: małe rozszerzenia dodające obsługę nowego modelu maszyny bez wydawania nowej wersji systemu. Dla wdrożeniowca była to jednocześnie ulga i utrapienie. Ulga, bo na całą flotę wystarczała jedna wersja systemu. Utrapienie, bo obraz instalacyjny przestawał być uniwersalny — trzeba było pilnować, żeby dla każdego modelu w firmie leżał w teczce systemowej właściwy enabler, i żeby aktualizacja systemu go nie usunęła.

System 7.5: wszystko i kuchenny zlew

System 7.5 wydano we wrześniu 1994 i jest to wydanie, które wchłonęło mnóstwo funkcji będących wcześniej shareware'em albo osobnymi produktami. Lista nowości brzmi jak spis treści katalogu narzędzi: **Apple Guide** (interaktywna pomoc prowadząca krok po kroku i podświetlająca realne elementy interfejsu — rozwiązanie wtedy nowatorskie i do dziś rzadko powtarzane), **Launcher**, **WindowShade** (zwijanie okna do paska tytułu, przejęte wprost od shareware'u), **Control Strip**, **Desktop Printing**, rozbudowany **Extensions Manager**, hierarchiczne menu Apple, skryptowalny Finder, przeciąganie tekstu i **Stickies**.

Jedna z aktualizacji tej linii jest historycznie ważna z zupełnie nietechnicznego powodu. **System 7.5.1** z marca 1995 był drobny, ale to w nim po raz pierwszy ekran startowy pokazał logo **„Mac OS”** zamiast napisu „Welcome to Macintosh”. Nazwa systemu zaczęła istnieć jako marka prawie dwa lata przed tym, zanim stała się oficjalna.

Mac OS 7.6, czyli skąd wzięła się nazwa „Mac OS”

Mac OS 7.6 wydano w styczniu 1997 i była to pierwsza wersja, która **oficjalnie nazywała się „Mac OS”**, a nie „System Software”. Powód tej zmiany jest czysto biznesowy i jest jedną z ładniejszych historii w całym materiale.

Apple licencjonowało wtedy swój system producentom klonów — o czym osobna sekcja dalej. Problem polegał na tym, że marka „System Software 7.5” była nierozłącznie zrośnięta z marką **Macintosh**, a klony nie mogły nazywać się Macintoshami, bo to była nazwa zastrzeżona przez Apple dla własnego sprzętu. Producent klona potrzebował więc nazwy produktu **oddzielnej od nazwy komputera** — takiej, którą można wydrukować na pudełku maszyny, która Macintoshem nie jest. Stąd „Mac OS”.

Mac OS 7.6.1 uruchomiony w emulatorze SheepShaver — ostatnie wydanie linii Systemu 7. Zwróć u

Mac OS 7.6.1 uruchomiony w emulatorze SheepShaver — ostatnie wydanie linii Systemu 7. Zwróć uwagę na nazwę „Mac OS”, oficjalną od wersji 7.6, pod którą ten sam system sprzedawano także z klonami. Zrzut ekranu: Apple; przytoczony w celu omówienia (prawo cytatu, art. 29 pr. aut.). Plik: en.wikipedia, non-free.

> **Kolejność zdarzeń jest tu warta uwagi.** Logo „Mac OS” pojawiło się na ekranie startowym już w System 7.5.1 w marcu 1995 — czyli w tym samym roku, w którym ruszyło licencjonowanie klonów. Oficjalna zmiana nazwy produktu przyszła dwa lata później. Marka wyprzedziła decyzję o dwa lata, co sugeruje, że była przygotowywana świadomie i z wyprzedzeniem, a nie wymyślona pod presją.

Mac OS 8: pretekst, ratunek i rozebrany Copland

Mac OS 8 wydano **26 lipca 1997** pod kryptonimem **„Tempo”** i sprzedano **ponad 1,2 miliona kopii w pierwsze dwa tygodnie**. Było to jedno z najbardziej udanych wydań pudełkowych w historii Apple — w momencie, w którym firma stała na finansowej krawędzi. Nie jest to przypadek: klienci kupowali system, bo wyglądał na dowód, że Apple jeszcze żyje.

Zawartość Mac OS 8 w dużej mierze pochodzi z **Coplanda**, czyli z anulowanego następcy, o którym osobna sekcja. Kiedy Ellen Hancock zamroziła projekt w sierpniu 1996, Apple nie wyrzuciło dorobku, tylko zaczęło przenosić pojedyncze technologie Coplanda do kolejnych wydań klasycznego systemu. Mac OS 8 jest pierwszym dużym beneficjentem tej operacji i dostał z niej:

- **Wygląd Platinum** — trójwymiarowy, szary motyw interfejsu, zaprojektowany pierwotnie jako skórka Coplanda.
- **Finder wielowątkowy i natywny dla PowerPC** — kopiowanie plików przestaje zamrażać interfejs. To pierwsze realne wejście wielowątkowości do powłoki i pierwszy moment, w którym Finder przestaje być programem 68k. Trzy lata i cztery miesiące po premierze PowerPC.
- **Menu kontekstowe** (Control-klik), **foldery sprężynujące**, konfigurowalne czcionki systemowe i kolory akcentów, karty okien na biurku.

Mac OS 8.1 z wyglądem Platinum. Zwróć uwagę na szary, trójwymiarowy motyw okien i przycisków,

Mac OS 8.1 z wyglądem Platinum. Zwróć uwagę na szary, trójwymiarowy motyw okien i przycisków, zaprojektowany pierwotnie jako skórka Coplanda. Zrzut ekranu: Apple; przytoczony w celu omówienia (prawo cytatu, art. 29 pr. aut.). Plik: en.wikipedia, non-free.

I rzecz najważniejsza z punktu widzenia historii firmy, o której szerzej w sekcji o klonach: **Mac OS 8 był narzędziem prawnym.** Licencje klonowe mówiły o „Systemie 7” i system pod numerem 8 formalnie ich nie obejmował.

Mac OS 8.1 i 8.5: HFS+ i koniec ery 68k

Mac OS 8.1 ze stycznia 1998 wprowadził **HFS+**, sprzedawany pod nazwą Mac OS Extended. Rozwiązał problem bloków alokacji przez 32-bitowe numerowanie, podniósł limit długości nazwy pliku do 255 znaków w Unicode i powiększył węzły B-drzewa katalogu. Był to zarazem **ostatni klasyczny Mac OS działający na procesorach 68k** — konkretnie na 68040. Czternaście lat linii 68000 kończy się w tym miejscu.

Mac OS 8.1 w emulatorze SheepShaver — wydanie, które wprowadziło HFS+ i jako ostatnie działało

Mac OS 8.1 w emulatorze SheepShaver — wydanie, które wprowadziło HFS+ i jako ostatnie działało na procesorach 68k. Zrzut ekranu: Apple; przytoczony w celu omówienia (prawo cytatu, art. 29 pr. aut.). Plik: en.wikipedia, non-free.

Mac OS 8.5 z października 1998 był pierwszym systemem wyłącznie dla PowerPC i skorzystał z tego natychmiast: usunięcie kodu 68k z wielu komponentów dało odczuwalny wzrost wydajności. Nowości: **Sherlock** — wyszukiwarka łącząca przeszukiwanie dysku, indeksowanie treści i wyszukiwanie w Internecie przez wtyczki — motywy wyglądu, **Application Switcher** jako odrywalny pasek, **Apple Data Detectors** rozpoznające adresy i daty w tekście, nowa ikonografia 32-bitowa i szybsze QuickDraw.

Mac OS 9: ostatni klasyk

Mac OS 9 wydano **23 października 1999**, aktualizacja kosztowała 99 dolarów, a Apple promowało go hasłem *„The Best Internet Operating System Ever”*. To jest czysty slogan marketingowy, nie opis techniczny, i podaję go wyłącznie dlatego, że dobrze oddaje, w którą stronę patrzyła wtedy firma.

Co faktycznie przyszło z dziewiątką:

- **Sherlock 2** — przebudowany interfejs, wyszukiwanie w „kanałach” (zakupy, wiadomości, ludzie). Mocno reklamowany i mocno krytykowany za interfejs stylizowany na odtwarzacz muzyczny.
- **Multiple Users** — obsługa wielu użytkowników bez dokupowania pakietu At Ease, w trzech trybach kont: Normal, Limited, Panels. To pierwsze konta użytkowników w klasycznym Mac OS, ale realizowane na poziomie powłoki: bez izolacji systemu plików i bez izolacji pamięci.
- **Keychain** — szyfrowany magazyn haseł. Jedna z niewielu technologii tego wydania, która przeżyła system i istnieje do dziś w macOS.
- **Software Update** — panel sterowania automatycznie pobierający aktualizacje z serwerów Apple. Pierwszy raz Mac aktualizuje się sam.
- **Carbon** — biblioteka API stanowiąca podzbiór klasycznego Toolboxa, przeznaczona do działania zarówno w Mac OS 9, jak i natywnie w Mac OS X.

> **Carbon jest w tej liście rzeczą najważniejszą.** Nie dlatego, że coś dawał użytkownikowi — bo nie dawał nic widocznego — tylko dlatego, że był ścieżką migracji dla Adobe, Microsoftu i Macromedii. Bez Carbona duzi producenci oprogramowania musieliby przepisać swoje pakiety od zera na zupełnie obce API, a wtedy Mac OS X nie miałby czym wystartować. Klasyczny Mac OS przekazał następcy nie tylko wygląd i nazwę, ale przede wszystkim zestaw wywołań, dzięki któremu Photoshop dało się przenieść w rozsądnym czasie.

Architektura i jej granice

To jest serce tego tekstu. Wszystko, co w klasycznym Mac OS wygląda dziwnie — konflikty rozszerzeń, ręczne przydzielanie pamięci, bomby, restart po jednym błędnym programie, ale też możliwość przerobienia dowolnego programu bez kodu źródłowego — wynika z kilku decyzji projektowych podjętych w latach 1982–1983 dla maszyny ze 128 kilobajtami pamięci i bez dysku twardego. Te decyzje były wtedy rozsądne. Problem polegał na tym, że przetrwały siedemnaście lat i nikomu nie udało się ich wymienić.

Brak ochrony pamięci i jedna przestrzeń adresowa

Klasyczny Mac OS nie miał ochrony pamięci. Wszystkie procesy — system, Finder, każda aplikacja, każde rozszerzenie, każdy sterownik — działały w **jednej, wspólnej przestrzeni adresowej**, bez jakiejkolwiek izolacji sprzętowej. Nie było pojęcia „pamięć należąca do procesu A”; była pamięć, a w niej umownie wyznaczone obszary, których wszyscy mieli nie ruszać.

Konsekwencje tej jednej właściwości są warte wyliczenia po kolei, bo dopiero razem dają obraz tego, jak wyglądała diagnostyka na tym systemie:

- Błędny wskaźnik w dowolnej aplikacji mógł nadpisać strukturę danych systemu, Findera albo innego, zupełnie niewinnego programu.
- Objawy awarii pojawiały się **daleko od przyczyny** — w czasie i w przestrzeni. Program A psuł pamięć o dziesiątej, program B wywalał się o jedenastej trzydzieści i to na programie B skupiało się śledztwo. Diagnostyka była w praktyce niemożliwa.
- Zamknięcie zawieszonej aplikacji **nie przywracało zdrowego stanu**, bo uszkodzenia, które zdążyła narobić, zostawały w pamięci. Jedynym pewnym lekarstwem był restart.
- Nie istniała granica bezpieczeństwa między programami. Model bezpieczeństwa tego systemu zakładał wprost, że wszystkie uruchomione programy są przyjazne i kompetentne.

> **Pamięć wirtualna to nie jest ochrona pamięci.** To najczęstsze nieporozumienie dotyczące Systemu 7 i warto je rozbroić raz na zawsze. Pamięć wirtualna Systemu 7 używała układu MMU do stronicowania zawartości pamięci na dysk, ale nadal mapowała wszystko do jednej przestrzeni adresowej. Dawała *więcej* pamięci, nie *bezpieczniejszą*. System potrafił czasem wykryć, że aplikacja pisze poza swoją partycją, ale tylko w części przypadków i tylko przy odpowiedniej konfiguracji MMU. To nie była izolacja procesów w żadnym sensie tego słowa.

Ochrona pamięci przyszła na Maca dopiero z jądrem Mach w Mac OS X w 2001 roku — **siedemnaście lat po premierze Macintosha** i pięć lat po tym, jak Copland, który miał ją

dostarczyć, został zamrożony. To jest, jak sądzę, najdłuższy udokumentowany przypadek odkładania tej samej poprawki architektonicznej w historii systemów osobistych, i jedyny, w którym poprawka ostatecznie przyszła razem z kupionym od kogoś innego systemem.

Wielozadaniowość kooperacyjna i `WaitNextEvent()`

To jest mechanizm, który definiował codzienne doświadczenie użytkownika klasycznego Maca bardziej niż cokolwiek innego — łącznie z wyglądem interfejsu.

Od MultiFindera z 1987 roku Mac OS potrafił trzymać w pamięci kilka aplikacji naraz, ale **system nie mógł ich wywłaszczyć**. Procesor przechodził do innego procesu wyłącznie wtedy, gdy bieżąca aplikacja sama go oddała. Apple nazywało to wprost *cooperative multitasking environment* i uzasadniało nazwę równie wprost: ponieważ płynne działanie wszystkich aplikacji zależy od ich współpracy.

Momentem oddania sterowania było wywołanie funkcji Menedżera Zdarzeń **`WaitNextEvent()`**, ewentualnie `EventAvail()`. Dokumentacja Apple mówiła to bez owijania: aplikacja może oddać kontrolę nad procesorem innym procesom wyłącznie w bardzo konkretnych momentach, mianowicie gdy wywoła jedną z tych dwóch funkcji. Sygnatura wyglądała tak:

```
Boolean WaitNextEvent(EventMask eventMask, EventRecord *theEvent,
                      UInt32 sleep, RgnHandle mouseRgn);
```

Dwa z tych parametrów warto omówić, bo pokazują, jak bardzo model kooperacyjny był wpisany w sposób pisania programów:

- **`sleep`** — liczba tyknięć zegara (tyknięcie to jedna sześćdziesiąta sekundy), na które aplikacja zrzeka się procesora, jeśli nie ma dla niej żadnych zdarzeń. Apple rekomendowało wartość niezerową, rzędu sześćdziesięciu tyknięć, dla aplikacji nierobiących nic w tle. Ustawienie `sleep = 0` było w praktyce głodzeniem wszystkich procesów działających w tle — i było to wyłącznie kwestią dobrej woli programisty.
- **`mouseRgn`** — region ekranu, poza który wyjście kursora generowało zdarzenie. Dzięki temu aplikacja mogła zmieniać kształt kursora bez odpytywania myszy w pętli, czyli bez zabierania procesora na coś, o czym system i tak wiedział.

A teraz to, co się działo, gdy program przestawał oddawać sterowanie. Jeśli aplikacja weszła w długą pętlę obliczeniową bez wywoływania `WaitNextEvent` — albo weszła w pętlę nieskończoną wskutek błędu — to następowało po kolei:

- kursor przestawał reagować albo zamierał w postaci zegarka;
- menu przestawały się rozwijać, bo obsługą menu zajmował się Menedżer Zdarzeń *tej* aplikacji;
- **wszystkie pozostałe aplikacje zamierały**, bo nigdy nie dostawały procesora;

- nie dało się przełączyć do Findera, żeby cokolwiek zrobić;
- system nie miał żadnego mechanizmu, żeby to przerwać.

Jedynym ratunkiem od Systemu 7 było **Command-Option-Escape** — wymuszone zamknięcie aplikacji pierwszoplanowej realizowane przez przerwanie sprzętowe. Rzecz w tym, że w systemie bez ochrony pamięci taka operacja zostawiała system w stanie nieokreślonym, i komunikat, który Apple przy tym wyświetlało, był wyjątkowo szczery: radził zapisać pracę w pozostałych programach i **zrestartować komputer**. Trudno o lepszą ilustrację tego, że producent sam wiedział, czego jego system nie potrafi.

> **Uczciwie o zaletach tego modelu.** Kooperacja miała realne przewagi i warto je wymienić, żeby nie wyszła karykatura. Aplikacja pierwszoplanowa dostawała praktycznie cały procesor, więc odczucie szybkości było znakomite. Nie było kosztu przełączania kontekstu przy każdym tyknięciu zegara. Nie było problemów z synchronizacją — Toolbox mógł nie być reentrantny, bo nikt nie wchodził w niego równolegle, co oszczędzało zarówno pamięć, jak i całą klasę błędów. Na maszynie z ośmioma megabajtami pamięci i procesorem taktowanym kilkunastoma megahercami kooperacja była po prostu tańsza. Do momentu, w którym przestała wystarczać.

Apple próbowało łatać ten model dwa razy. **Thread Manager** z 1993 roku, włączony potem do System 7.5, dał wielowątkowość *wewnątrz* aplikacji — również kooperacyjną, czyli wątki musiały same oddawać sobie sterowanie. **Multiprocessing Services** z 1996 roku dały już zadania wywłaszczające, ale z ograniczeniem, które przekreślało większość zastosowań: zadanie MP **nie mogło wywoływać Toolboxa**. Nie rysowało, nie dotykało plików, nie alokowało pamięci. Nadawało się do liczenia i do niczego, co widać na ekranie.

Uchwyty i bloki relokowalne: dlaczego trzeba było blokować uchwyt

Oryginalny Macintosh miał 128 kilobajtów pamięci i nie miał pamięci wirtualnej. Żeby w tak małej przestrzeni dało się w ogóle sensownie gospodarować, Apple wprowadziło **bloki relokowalne**, adresowane przez **uchwyty** (handles). Jest to mechanizm, który dla programisty wychowanego na `malloc` wygląda na komplikację bez powodu, a był jedyną rzeczą, która sprawiała, że sterta o rozmiarze kilkudziesięciu kilobajtów nie rozsypywała się w kilka minut pracy.

Definicja jest prosta: **uchwyt to wskaźnik na wskaźnik**. W języku C zapisywano to jako `typedef char **Handle;`. Uchwyt wskazuje na pozycję w nierelokowalnej tablicy **wskaźników głównych** (master pointers), a dopiero ten wskaźnik główny wskazuje na faktyczne dane. Menedżer Pamięci może więc dowolnie przesunąć blok danych po stercie, aktualizując przy tym jeden jedyny wskaźnik główny — a wszystkie uchwyty trzymane w programie pozostają ważne, bo wskazują na pozycję w tablicy, która się nie ruszyła.

Po co to było: **kompaktowanie sterty**. Kiedy brakowało pamięci, Menedżer Pamięci przesuwał wszystkie ruchome bloki „do siebie”, scalając wolne miejsce w jeden ciągły kawałek. Bez tego sterta stu kilkudziesięciu kilobajtów fragmentowałaby się do bezużyteczności w ciągu kilkunastu minut normalnej pracy.

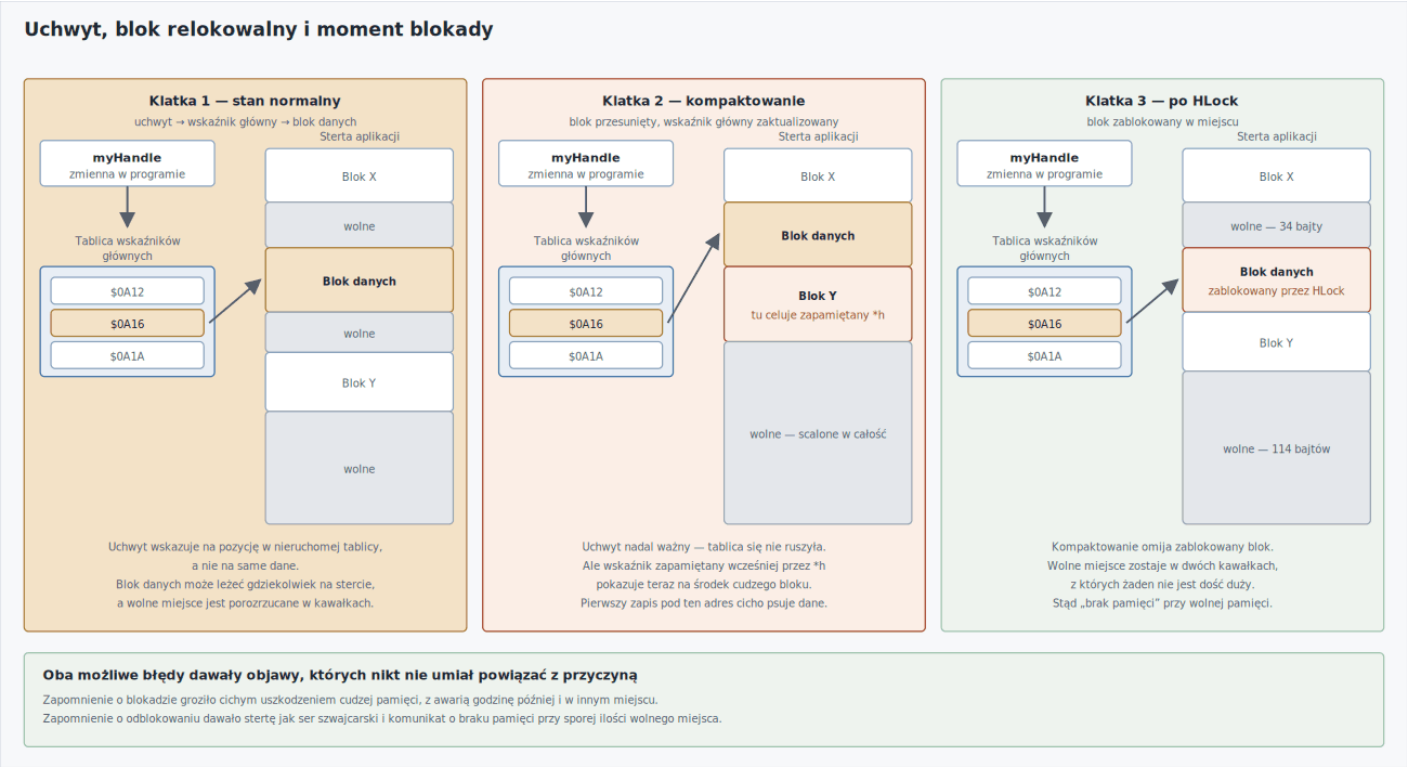
Cena była następująca: **wskaźnik uzyskany przez wyłuskanie uchwytu przestawał być ważny w nieprzewidywalnym momencie**. Dokumentacja mówiła to wprost — wskaźniki do danych w blokach relokowalnych nie mają gwarancji ważności pomiędzy wywołaniami.

W praktyce wyglądało to tak, i jest to absolutny klasyk gatunku. Programista pisał `*myHandle`, żeby dostać wskaźnik do danych. Potem wywoływał dowolną funkcję Toolboxa mogącą zaalokować pamięć — a mogła praktycznie każda: `NewWindow`, `GetResource`, nawet `DrawString`. Menedżer Pamięci przy okazji tej alokacji kompaktował stertę. Blok się przesuwał. A stary wskaźnik, wciąż trzymany w zmiennej lokalnej, wskazywał teraz na środek zupełnie innego obiektu — i pierwszy zapis pod ten adres cicho psuł cudze dane. Awaria następowała gdzie indziej i godzinę później.

Lekarstwem było **`HLock(myHandle)`**, czyli zablokowanie bloku w miejscu na czas pracy ze wskaźnikiem, i **`HUnlock`** po zakończeniu. Wokół tej pary narosł zestaw funkcji pomocniczych, którego znajomość odróżniała programistę doświadczonego od początkującego:

- `HGetState` i `HSetState` — bezpieczniejsze niż ślepe `HUnlock`, bo zapamiętują i odtwarzają poprzedni stan bloku zamiast go odblokowywać; dzięki temu nie odblokowuje się przypadkiem bloku, który ktoś inny zablokował z własnego powodu.
- `MoveHHi` — przesunięcie bloku na górę sterty *przed* zablokowaniem, żeby zablokowany blok nie wylądował w środku i nie zafragmentował sterty.
- `MaxApplZone()` i `MoreMasters()` — dwa wywołania, od których zaczynał się każdy podręcznik programowania na Maca. To drugie prealokowało bloki wskaźników głównych, żeby nie powstawały później w losowych miejscach sterty, znów ją fragmentując.

I teraz sedno: **zapomnienie o blokadzie groziło uszkodzeniem pamięci, a zapomnienie o odblokowaniu — fragmentacją**. Oba błędy dawały objawy, których użytkownik nie umiał powiązać z ich przyczyną. Zablokowany blok w środku sterty tworzył nieusuwalną barierę, wokół której kompaktowanie nie działało. Po pewnym czasie pracy sterta aplikacji wyglądała jak ser szwajcarski i program meldował brak pamięci przy sporej ilości wolnego miejsca — bo wolne miejsce było, tylko w dziesięciu kawałkach, z których żaden nie był dość duży.



Rysunek 3. *Trzy stany tej samej sterty pokazują, dlaczego blokada uchwytu była konieczna i dlaczego jednocześnie była kosztowna: bez niej wskaźnik cicho psuje cudze dane, z nią zablokowany blok zostaje barierą, o którą rozbija się kompaktowanie.*

Strefy pamięci i okno Informacje: użytkownik jako administrator pamięci

Pamięć klasycznego Maca dzieliła się na **strefy** (heap zones). Podstawowe były dwie: **sterta systemowa** i **sterta aplikacji**.

Każda aplikacja pod MultiFinderem dostawała **partycję** o stałym rozmiarze. Wewnątrz tej partycji układ był następujący:

- **Stos aplikacji** — rośnie w dół od górnej granicy partycji; trzyma zmienne lokalne i ramki wywołań.
- **Świat A5** — obszar adresowany przez rejestr A5 procesora: globalne zmienne QuickDraw, globalne zmienne aplikacji oraz **tablica skoków** z jedną pozycją na każdą procedurę wywoływaną spoza swojego segmentu kodu.
- **Sterta aplikacji** — rośnie w górę od dolnej granicy partycji; trzyma segment `CODE` 1, struktury danych, zasoby i doładowywane w miarę potrzeby segmenty kodu.

Rozmiar partycji brał się z **zasobu** `SIZE` w pliku aplikacji: wartości preferowanej i minimalnej. Menedżer Procesów próbował przydzielić wartość preferowaną; gdy pamięci brakowało, schodził do minimalnej; gdy i tej nie było, aplikacja po prostu się nie uruchamiała z komunikatem o braku pamięci.

I tu wchodzi najbardziej charakterystyczna rzecz w całej tej architekturze. **Użytkownik mógł te wartości zmienić ręcznie.** Zaznaczał ikonę aplikacji w Finderze, wybierał **Get Info** (Command-I, w spolszczonym żargonie po prostu „Informacje”) i w sekcji „Memory Requirements” wpisywał kilobajty w trzy pola: **Suggested** — tylko do odczytu, wprost z zasobu `SIZE` — oraz **Minimum** i **Preferred**, które były edytowalne. Technicznie rzecz biorąc, użytkownik edytował w tym momencie zasób w cudzym pliku wykonywalnym, i robił to przez okno dialogowe dostarczone przez producenta systemu.

Konsekwencje tego modelu, wypisane z perspektywy człowieka, który taką flotę obsługuje:

- **Przydział był statyczny.** Aplikacja, której dano osiem megabajtów, trzymała osiem megabajtów przez cały czas działania, nawet jeśli faktycznie używała jednego. Pamięci nie dało się współdzielić między programami ani oddać systemowi na chwilę. Prowadziło to do jawnie nieefektywnej alokacji przy każdej konfiguracji poza idealną.
- **Użytkownik musiał wiedzieć, ile dać.** Ustawianie przydziałów nie było egzotycznym tuningiem dla zaawansowanych, tylko codzienną czynnością — zwłaszcza w DTP, gdzie Photoshop z dużym plikiem potrafił zjeść wszystko, co mu się dało, i poprosić o więcej.
- **Objaw „za mało pamięci” był mylący.** Komunikat o braku pamięci wyświetlany przez aplikację oznaczał, że skończyła się *jej partycja* — a nie pamięć maszyny. Użytkownik patrzył na okno „About This Macintosh” pokazujące czterdzieści wolnych megabajtów i nie rozumiał, dlaczego program twierdzi, że pamięci brakuje. Jest to, moim zdaniem, najtrudniejszy do wytłumaczenia koncepcyjnie element całego systemu — trudniejszy niż uchwyty, bo uchwytów użytkownik nie widział, a tego komunikatu nie dało się nie zobaczyć.
- **Okno „About This Macintosh”** w menu Apple pokazywało paski zajętości dla każdej uruchomionej aplikacji i dla systemu. Był to menedżer zadań epoki — z tą różnicą, że wyłącznie do oglądania. Nic nie dało się w nim zrobić.

> Ten model jest najlepszą ilustracją tezy całego tekstu: **klasyczny Mac OS przerzucał na człowieka zadania, które w systemach z ochroną pamięci i dynamiczną alokacją wykonuje jądro.** Człowiek robił to gorzej niż jądro, ale robił to *świadomie* — i dlatego rozumiał swoją maszynę lepiej niż przeciętny użytkownik dzisiaj. To nie jest argument za takim projektem. To jest obserwacja o tym, co się przy okazji straciło.

Menedżer Zasobów i widelec zasobów

Plik w klasycznym Mac OS składał się z **dwóch strumieni**: **widelca danych** (data fork) i **widelca zasobów** (resource fork). Jest to jedna z tych rzeczy, które wyglądają na kaprys, a okazują się jednym z najdalej idących pomysłów tego systemu.

Widelec danych to zwykły, nieustrukturyzowany, sekwencyjny strumień bajtów — dokładnie to, co w innych systemach nazywa się po prostu plikiem. Widelec zasobów to natomiast **ustrukturyzowana baza danych z dostępem swobodnym**, złożona z trzech części: nagłówka zasobów z metadanymi, **mapy zasobów** z pozycjami poszczególnych elementów i wreszcie

samych danych. Mapa pozwala sięgnąć po dowolny element na podstawie jego identyfikatora albo nazwy, bez czytania całości.

Każdy zasób miał **typ** — czterobajtowy identyfikator `OSType`, na przykład `ICN#`, `MENU`, `WIND`, `DLOG`, `STR#`, `CODE`, `PICT`, `snd ` — szesnastobitowy identyfikator ze znakiem oraz opcjonalną nazwę. Menedżer Zasobów układał otwarte widelce w **stos**, w którym zasoby lokalne przesłaniały systemowe. Dzięki temu aplikacja mogła podstawić własną ikonę, czcionkę czy kursor w miejsce standardowych, nie modyfikując niczego w systemie — po prostu miała je wyżej na stosie.

Do edycji tego wszystkiego Apple wydawało za darmo narzędzie **ResEdit**, z graficznymi edytorami dla znanych typów zasobów: ikony rysowało się piksel po pikselu, dialogi układało myszą, menu i listy tekstów edytowało w tabelce. Istniała też alternatywa komercyjna, **Resorcerer**, bardziej rozbudowana. Kluczowa właściwość ResEdit z punktu widzenia tej strony brzmi: **pozwalał nieprogramistom modyfikować cudze aplikacje bez kodu źródłowego**.

> **Lokalizacja przez podmianę zasobów była oficjalnym modelem Apple, a nie hackiem.** Skoro wszystkie teksty interfejsu leżały w zasobach `STR#`, `MENU`, `DLOG` i `DITL`, to przetłumaczenie programu nie wymagało ani rekompilacji, ani dostępu do kodu: wystarczyło otworzyć plik w ResEdit, pozmieniać napisy i poszerzyć pola dialogów, bo polski i niemiecki są dłuższe od angielskiego. To był zalecany przez Apple model internacjonalizacji. Wrócę do tego w wątku polskim, bo tłumaczy on rzecz, której inaczej nie da się wytłumaczyć: dlaczego w Polsce powstał oddolny ekosystem polonizatorów, skoro Apple nigdy nie wydało polskiej wersji systemu.

Poza światem Maca widelce były jednak czystym utrapieniem, bo FAT, NFS i protokoły internetowe nie znają pliku o dwóch strumieniach. Obejścia mnożyły się przez całe lata dziewięćdziesiąte:

- **MacBinary** i **BinHex** — oba widelce zapakowane w jeden strumień; stąd pliki z rozszerzeniami `.bin` i `.hqx` na ówczesnych serwerach FTP.
- **AppleDouble** — widelec zasobów w osobnym, ukrytym pliku z przedrostkiem `._`. Są to dokładnie te same pliki, które współczesny macOS do dziś zostawia na pendrive'ach i które irytują wszystkich, którzy nie wiedzą, skąd się biorą.
- **AFP** — obsługa natywna, bez obejść; alternatywne strumienie NTFS przez SMB jako rozwiązanie po stronie serwera Windows.

> **Konsekwencja dla administratora była poważna: kopia zapasowa zrobiona narzędziem nieświadomym widelców niszczyła pliki.** Skopiowanie aplikacji macowej zwykłym FTP w trybie binarnym dawało plik z pustym widelcem zasobów — czyli aplikację, która się nie uruchamiała, przy zachowanym rozmiarze i dacie. Awaria była cicha: backup wyglądał na wykonany, katalog na miejscu, a odtworzone pliki nie działały. Była to codzienność wsparcia technicznego tamtych lat i jest to wzorzec awarii, który warto pamiętać przy każdej migracji danych między systemami o różnych modelach metadanych.

Pułapki systemowe i łatanie jako technika oficjalna

Tu dochodzimy do decyzji, która wyjaśnia największą liczbę zjawisk w tym systemie — i dobrych, i złych.

Toolbox i System Maca wywoływano nie przez zwykłe skoki podprogramów, tylko przez **pułapki procesora 68000**, konkretnie przez **instrukcje A-line**: szesnastobitowe słowa zaczynające się od bitów `1010`. Procesor 68000 traktował takie słowo jako nieznaną instrukcję i generował wyjątek, który system przechwytywał. Powtarzane twierdzenie, że Motorola zarezerwowała tę grupę słów właśnie po to, by dało się rozszerzać zestaw instrukcji programowo, jest prawdopodobne — ale nie znalazłem na nie źródła i podaję je jako relację, nie jako ustalenie.

Trap Dispatcher brał numer pułapki z młodszych bitów słowa, używał go jako indeksu w **tablicy pułapek** i skakał pod zapisany tam adres. I teraz zdanie, które trzeba przeczytać uważnie, bo od niego zależy reszta: **tablica pułapek leżała w pamięci operacyjnej, nie w ROM-ie.**

Skoro tablica leży w pamięci zapisywalnej i ma znane położenie, to dowolny kod może podmienić w niej adres — czyli przechwycić i zastąpić dowolną funkcję systemową. Apple nie tylko o tym wiedziało; Apple udostępniło do tego **oficjalne API**: `NGetTrapAddress()` i `NSetTrapAddress()`. Kanoniczny wzorzec łatki wyglądał tak:

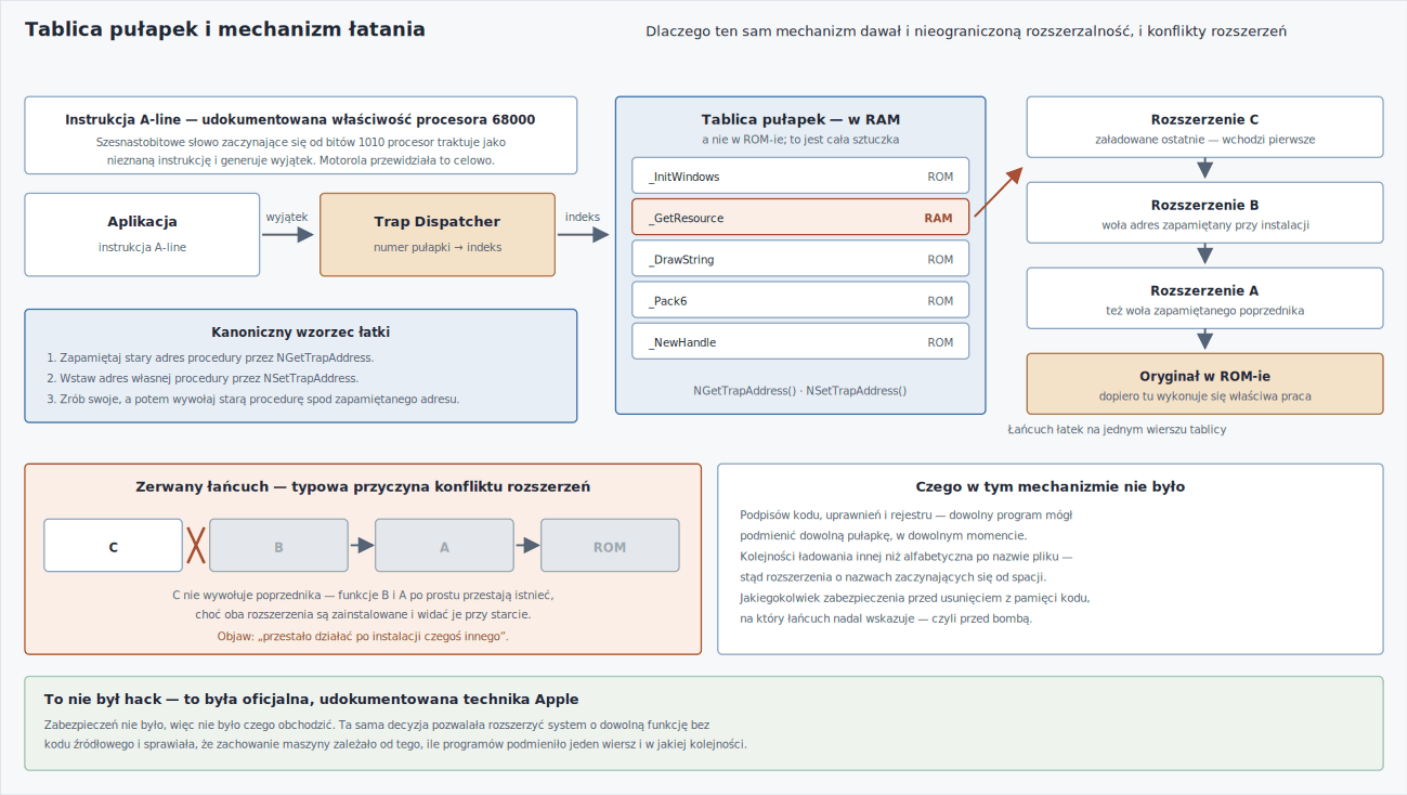
1. Zapamiętaj stary adres procedury przez `NGetTrapAddress`.
2. Wstaw adres własnej procedury przez `NSetTrapAddress`.
3. W swojej procedurze zrób to, co masz do zrobienia, a potem **wywołaj starą procedurę** spod zapamiętanego adresu.

To się nazywało *patching a trap* i było **oficjalną, udokumentowaną techniką Apple**, a nie hackiem obejścia zabezpieczeń — zabezpieczeń nie było, więc nie było czego obchodzić. Apple używało tej techniki samo, i to intensywnie: plik `System` przy starcie instalował dziesiątki łatek na procedury zaszyte w ROM-ie. Dlatego stary Macintosh z nowym Systemem potrafił rzeczy, których jego własny ROM w ogóle nie znał. Wracamy tym samym do rysunku 1: system był nakładką, a mechanizm nakładania musiał być publiczny.

Łańcuchowanie łatek działało i było normą. Kilka rozszerzeń mogło załatać tę samą pułapkę, tworząc łańcuch wywołań, w którym każde kolejne zapamiętywało adres „poprzednika” — który sam mógł już być czyjąś łatką — i wołało go na końcu swojej pracy. Przy sprawnie napisanych rozszerzeniach mechanizm był elegancki: system dawał się rozbudowywać w nieskończoność, bez źródeł i bez zgody producenta.

> **I tu leży źródło wszystkich konfliktów rozszerzeń.** Jeżeli dwa rozszerzenia łatają tę samą pułapkę i jedno z nich zapomni wywołać poprzednika, funkcje drugiego przestają istnieć. Jeżeli zmieni argumenty albo zawartość rejestrów w sposób, którego drugie się nie

> spodziewa — będzie awaria. Jeżeli zostanie usunięte z pamięci, a łańcuch nadal wskazuje na jego kod — nastąpi skok pod nieistniejący adres i natychmiastowa bomba. Jeżeli wreszcie założy, że jest pierwsze w łańcuchu, zachowanie systemu zacznie zależeć od kolejności ładowania plików. **Nikt nie kontrolował, kto łata.** Nie było podpisów, uprawnień ani rejestru — dowolny program mógł podmienić dowolną pułapkę, w dowolnym momencie.



Rysunek 4. *Ten sam mechanizm, który pozwalał rozszerzyć system o dowolną funkcję bez kodu źródłowego, powodował, że zachowanie maszyny zależało od tego, ile programów podmieniło jeden wiersz tablicy i w jakiej kolejności.*

Rozszerzenia systemu i konflikty rozszerzeń

Rozszerzenia — *extensions* — to były pliki zawierające zasób typu `INIT`, stąd potoczna nazwa **INIT-y**, używana jeszcze długo po tym, jak System 7 zmienił oficjalną nazwę. Były to porcje kodu ładowane podczas startu systemu, które rozszerzały jego możliwości **przez bezpośrednie łatanie wywołań systemowych** — czyli dokładnie przez mechanizm z poprzedniej sekcji. Kiedy aplikacja wykonywała wywołanie systemowe, rozszerzenie je przechwytywało, ewentualnie modyfikowało i przekazywało dalej.

Kolejność ładowania była ustalana alfabetycznie, według porządku ASCII, w obrębie każdego podfolderu teczki systemowej, a same foldery przetwarzano w ustalonej sekwencji: najpierw `Extensions`, potem `Control Panels`, potem elementy leżące w korzeniu teczki systemowej — przy

czym szczegóły tej sekwencji zależały od wersji systemu. Najważniejsze jest jednak to, co z tego wynikało: **użytkownik mógł zmienić kolejność ładowania rozszerzeń, po prostu zmieniając nazwę pliku w Finderze**.

Start systemu objawiał się **rządkiem ikon na dole ekranu** — każde rozszerzenie rysowało swoją w momencie załadowania. Przy typowej instalacji z połowy lat dziewięćdziesiątych było ich od trzydziestu do sześćdziesięciu, w dwóch albo trzech wierszach. Pasek był zarazem paskiem postępu i podstawowym narzędziem diagnostycznym: przy zawieszeniu patrzyło się, **która ikona pojawiła się jako ostatnia**, i to od niej zaczynało się śledztwo.

Konflikt rozszerzeń był zjawiskiem tak powszechnym, że doczekał się nazwy własnej i osobnej dyscypliny wsparcia technicznego. Objawy: zawieszenie przy starcie, bomba w losowym momencie pracy, niedziałająca drukarka, funkcja, która zniknęła bez śladu. Metody diagnostyki układały się w wyraźną drabinę:

- **Start z wyłączonymi rozszerzeniami** — przytrzymanie klawisza **Shift** podczas startu, od Systemu 7. Pierwszy krok każdej diagnozy: jeśli z wciśniętym Shiftem maszyna działa, winne jest któreś rozszerzenie i wiadomo, gdzie szukać.
- **Extensions Manager** — od System 7.5 wbudowany w system, wywoływany przytrzymaniem **spacji** przy starcie. Pozwalał włączać i wyłączać pojedyncze rozszerzenia oraz zapisywać całe zestawy, co przy flocie maszyn było jedyną formą zarządzania konfiguracją.
- **Wyszukiwanie binarne** — metoda ręczna i żmudna: wyłącz połowę rozszerzeń, zrestartuj, sprawdź, podziel dalej. Przy sześćdziesięciu rozszerzeniach to jakieś sześć restartów — o ile problem był powtarzalny, co bardzo często nie było prawdą, bo objawy zależały od kolejności zdarzeń.
- **Conflict Catcher** firmy Casady & Greene, autorstwa Jeffreya Robbina — najsłynniejsze narzędzie tej kategorii. Automatyzował wyszukiwanie binarne, prowadząc użytkownika przez kolejne restarty i wskazując winowajcę. Miał bazę opisów znanych rozszerzeń, dzięki której użytkownik po raz pierwszy mógł się w ogóle dowiedzieć, *co* robi konkretny plik w jego teczce systemowej. Zarządzał też kolejnością ładowania i wykrywał uszkodzone pliki. Konkurentem był **Now Startup Manager**.

> **Przypis, który sam się prosi o wpisanie.** Jeffrey Robbin, autor Conflict Catchera, napisał później SoundJam MP — program, który Apple kupiło i przemianowało na **iTunes**. Człowiek, który zbudował najlepsze narzędzie do rozplątywania bałaganu w klasycznym Mac OS, skończył budując aplikację, która przez dekadę definiowała firmę.

Z perspektywy administratora floty najważniejsze zdanie o rozszerzeniach brzmi tak: **dwie maszyny o identycznym sprzęcie i identycznej wersji systemu mogły zachowywać się inaczej**, bo miały inny zestaw i inną kolejność rozszerzeń. Standaryzacja obrazu instalacyjnego była jedyną możliwą obroną — i padała w momencie, w którym użytkownik cokolwiek zainstalował, bo instalatory wrzucały swoje pliki do teczki systemowej bez pytania i bez rejestrowania, co zmieniły.

Przejście na PowerPC (1994) i emulator, który żył trzynaście lat

2 października 1991 Apple, IBM i Motorola zawiązały sojusz znany jako **AIM**, którego owocem była architektura **PowerPC**. Pierwsze Power Macintoshe — modele 6100/60, 7100/66 i 8100/80 — trafiły do sprzedaży **14 marca 1994** z systemem **System 7.1.2**.

Emulator 68k

Apple wbudowało w system **emulator 68k** działający na PowerPC. Pierwszą jego wersję napisał **Gary Davidian**, pierwotnie celując zresztą w procesor Motorola 88000 — a więc jeszcze przed zwrotem firmy ku PowerPC. Późniejszą wersję, z **rekompilacją dynamiczną**, stworzył **Eric Traut**.

Rekompilacja dynamiczna polegała na tłumaczeniu często wykonywanych fragmentów kodu 68k na sekwencje instrukcji PowerPC i buforowaniu wyniku, żeby nie tłumaczyć tego samego dwa razy. Efekt tej techniki był taki, że emulowany kod potrafił działać **szybciej niż na prawdziwym sprzęcie 68k**: na odpowiednio szybkim PowerPC emulowany program 68k bił wydajnością Quadrę. Jest to sytuacja, która brzmi paradoksalnie, a wynika po prostu z różnicy pokoleń krzemu — emulacja kosztuje, ale kosztuje mniej, niż wynosiła różnica zegarów.

Mixed Mode Manager, UPP i tłuste binaria

Kod 68k i kod PowerPC musiały się nawzajem wywoływać, i to w obie strony: aplikacja PowerPC wołała zasób systemowy będący kodem 68k, a rozszerzenie 68k łatało pułapkę wołaną przez kod PowerPC. Apple wprowadziło do tego **Mixed Mode Manager** — warstwę przełączającą tryb wykonania przy każdym przekroczeniu granicy architektur.

Mechanizmem były **Universal Procedure Pointers**, w skrócie UPP. UPP nie był zwykłym wskaźnikiem: wskazywał na strukturę danych zawierającą specjalną instrukcję pułapki oraz flagi określające architekturę zestawu instrukcji wywoływanego kodu. Dzięki temu można było mieszać kod obu architektur w sposób przezroczysty dla strony wołającej — która nie musiała wiedzieć, czym jest to, co woła.

Koszt tego rozwiązania był realny i odczuwalny: **każde przejście przez granicę architektur kosztowało**. Wywołanie przez aplikację PowerPC procedury systemowej, która nadal była kodem 68k, oznaczało przełączenie do emulatora i z powrotem. Dlatego aplikacja reklamowana jako „natywna” bywała wolniejsza od oczekiwań — nie dlatego, że była źle napisana, tylko dlatego, że połowa tego, co wołała, natywna nie była.

Tłuste binaria (fat binaries) to był pojedynczy plik aplikacji zawierający **dwa kompletne zestawy kodu**: 68k w zasobach `CODE` i PowerPC w formacie PEF. System uruchamiał wariant pasujący do procesora, a ceną był mniej więcej dwukrotny rozmiar pliku. Powstały nawet narzędzia zwane *fat strippers*, które wycinały niepotrzebną połowę — przy dyskach kilkusetmegabajtowych była to oszczędność, o którą warto było zawalczyć.

> **Ten sam wzorzec Apple powtórzyło potem dwukrotnie.** Universal Binary przy przejściu z PowerPC na Intela w 2006 roku, z emulatorem Rosetta, i Universal 2 przy przejściu z Intela na Apple Silicon w 2020 roku, z Rosettą 2. Trzy pokolenia tej samej metodyki: tłuste binarium plus emulator plus okres przejściowy. Pierwsze podejście — to z 1994 roku — było najdłuższe i najbardziej niekompletne, a firma najwyraźniej wyciągnęła z niego wnioski, bo kolejne dwa przeszły znacznie sprawniej. Przejście na Intela wraz z pierwszą Rosettą opisuje tekst **Mac OS X: przejście**, a przejście na Apple Silicon i Rosettę 2 — **macOS dzisiaj**.

Ile systemu zostało kodem 68k i jak długo

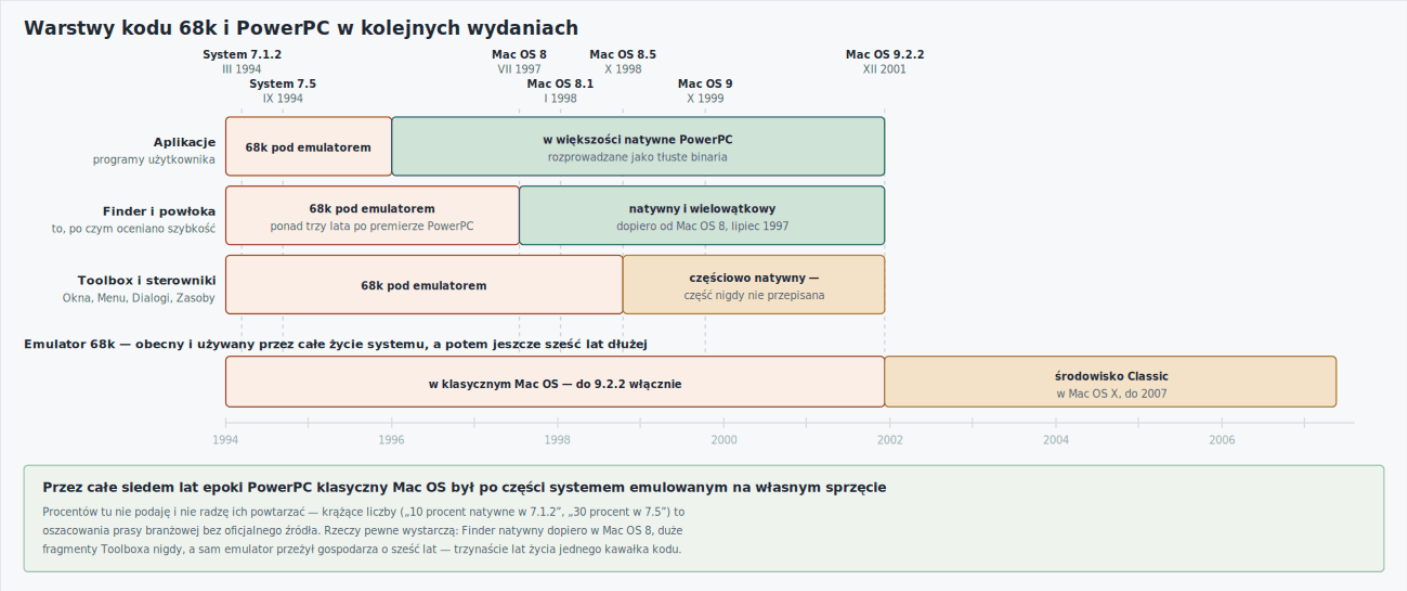
To jest najbardziej zaskakująca część tej historii i warto ją podać ostrożnie, bo krąży wokół niej sporo liczb bez pokrycia.

> **Procentów nie podaję i radzę ich nie powtarzać.** Krążą szacunki w rodzaju „około 10 procent systemu było natywne w 7.1.2”, „w System 7.5 około 30 procent”, „w Mac OS 8.5 większość, ale nie całość”. Nie znalazłem żadnego oficjalnego, liczbowego oświadczenia Apple na ten temat — wszystkie te liczby to oszacowania prasy branżowej i deweloperów, powtarzane potem bez źródła. Nie ma sensu ich przepisywać, tym bardziej że da się powiedzieć rzeczy pewne, które są ciekawsze od procentów.

Rzeczy pewne są następujące:

- **Finder pozostał aplikacją 68k aż do Mac OS 8**, czyli do lipca 1997 roku. Przez **ponad trzy lata** po premierze PowerPC najbardziej widoczny element całego systemu — ten, w którym użytkownik spędzał najwięcej czasu i po którym oceniał szybkość maszyny — działał w emulacji. Natywny, wielowątkowy Finder przyszedł dopiero z wydaniem, które Apple sprzedało w 1,2 miliona egzemplarzy w dwa tygodnie.
- **Duże fragmenty Toolboxa** — Menedżer Okien, Menedżer Menu, Menedżer Dialogów, Menedżer Zasobów — pozostawały kodem 68k bardzo długo, a częściowo aż do końca linii.
- **Mac OS 8.5 z 1998 roku** dopiero usunął kod 68k z na tyle wielu komponentów, że dało to odczuwalny wzrost wydajności — i było to zarazem pierwsze wydanie wymagające PowerPC, czyli pierwsze, które mogło sobie na taką operację pozwolić.
- **Mac OS 9.2.2 z grudnia 2001** — nawet w ostatniej wersji klasycznego Mac OS emulator 68k był obecny i używany. Pewne fragmenty systemu i sterowników nigdy nie zostały przepisane.

> **Wniosek, który trudno przecenić:** przez całe siedem lat epoki PowerPC — od 1994 do 2001 — **klasyczny Mac OS był po części systemem emulowanym na własnym, natywnym sprzęcie**. A emulator przeżył jeszcze samego gospodarza: był potrzebny w środowisku Classic w Mac OS X i pracował aż do 2007 roku, czyli do końca wsparcia Classic na sprzęcie PowerPC. Trzynaście lat życia jednego emulatora, z czego ostatnie sześć w systemie, dla którego nie był pierwotnie pisany.



Rysunek 5. *Migracja na PowerPC nie była zdarzeniem, tylko siedmioletnim procesem o nierównym tempie — i najbardziej widoczna część systemu przeszła na natywny kod jako jedna z ostatnich, ponad trzy lata po premierze nowego procesora.*

Paradoks Mac OS 8.6

Od pierwszych Power Maców pod klasycznym systemem siedziała cienka warstwa zarządzająca przerwaniami i kontekstem procesora, zwana **nanokernelem**. W **Mac OS 8.6** z maja 1999 roku została ona rozbudowana tak, by obsługiwać **Multiprocessing Services** w wersji 2.x — czyli prawdziwe, **wywłaszczające** zadania, działające również na maszynach wieloprocesorowych.

Powstała w ten sposób sytuacja, która jest moim zdaniem najdziwniejszym układem architektonicznym w całej historii systemów osobistych. **Pod kooperacyjnym, jednoprzestrzeniowym systemem operacyjnym pracował wywłaszczający mikro-scheduler** — z prawdziwym przełączaniem zadań, z prawdziwym planowaniem czasu procesora, z obsługą wielu procesorów fizycznych. A klasyczne aplikacje nie miały do niego żadnego dostępu.

Z punktu widzenia nanokernela **cały klasyczny system — Finder, wszystkie aplikacje, wszystkie rozszerzenia, cała ta kooperacyjna konstrukcja — był jednym zadaniem**. Jednym. Nanokernel przełączał się między tym jednym wielkim zadaniem a wydzielonymi zadaniami MP, które, jak już wiemy z sekcji o wielozadaniowości, nie mogły wołać Toolboxa, więc nie mogły nic narysować, zapisać ani zaalokować.

> **Morał, który wychodzi poza historię Apple.** Wywłaszczanie na dole stosu nie daje wywłaszczania na górze, jeśli cała warstwa aplikacyjna została zaprojektowana bez niego. Nie da się dołożyć izolacji i planowania pod spodem systemu, który zakłada, że wszystko dzieje się sekwencyjnie w jednej przestrzeni adresowej — bo ten system pozostanie jednym zadaniem. To jest dokładnie ten sam wniosek, który Apple wyciągnęło przy Coplandzie i przy środowisku Classic w Mac OS X, gdzie cały Mac OS 9 był, znowu, jednym procesem jądra Mach. Trzy razy ten sam układ, trzy razy ta sama granica.

Anulowane projekty: cmentarz następców

Apple podjęło co najmniej cztery poważne próby zastąpienia klasycznego Mac OS i przegrało wszystkie cztery. Ten rozdział jest, moim zdaniem, najbardziej pouczającą częścią całej historii — bardziej niż opis samej architektury, bo architekturę da się opisać, a wnioski z serii nieudanych przepisań trzeba wyciągnąć samemu.

Copland (1994–1996)

Copland to system operacyjny rozwijany przez Apple w latach 1994–1996, nigdy niewydany komercyjnie. Nazwa pochodzi od kompozytora Aarona Coplanda, zgodnie z przyjętą wtedy konwencją nazewniczą „kompozytorzy amerykańscy” — następca miał nazywać się Gershwin. Pierwotnie planowano go jako System 8, potem jako Mac OS 8.

Co miał mieć:

- **Ochronę pamięci** — każda aplikacja we własnej przestrzeni adresowej.
- **Wielozadaniowość wywłaszczającą** z prawdziwym planistą, a nie z dobrą wolą aplikacji.
- **Mikrokernel** z serwerami systemowymi nad nim.
- **Zgodność wsteczną** — stare aplikacje miały działać kooperacyjnie w wydzielonym „niebieskim polu” (blue box). Jest to dokładnie ta idea, którą zrealizowano później jako środowisko Classic w Mac OS X, tyle że na innym jądrze.
- Nowy, konfigurowalny wygląd — trafił ostatecznie do Mac OS 8 jako Platinum — i nowy Finder.
- Obsługę **OpenDoc**, **QuickDraw GX** i **SOM**.

Okno otwierania pliku z wersji rozwojowej Coplanda. Zwróć uwagę na nowy wygląd interfejsu — to z

Okno otwierania pliku z wersji rozwojowej Coplanda. Zwróć uwagę na nowy wygląd interfejsu — to z tego projektu wyszedł motyw, który trafił potem do Mac OS 8 jako Platinum. Zrzut ekranu: Apple;

przytoczony w celu omówienia (prawo cytatu, art. 29 pr. aut.). Plik: en.wikipedia, non-free.

Kalendarium upadku jest krótkie i czytelne:

Data	Zdarzenie
III 1994	Oficjalny start prac
V 1995	Demonstracja wczesnych komponentów na konferencji WWDC
VIII 1995	David Nagel ogłasza cel: wydanie w połowie 1996 roku
V 1996	Gil Amelio zapowiada wydanie pod koniec roku
VIII 1996	Developer Release 0 trafia do deweloperów i okazuje się bezużyteczny — sypie się samoczynnie
VIII 1996	Ellen Hancock zamraża prace; projekt przestaje być planowanym produktem

Dlaczego padł: jest to podręcznikowy przykład **rozrostu zakresu** i złego zarządzania. Relacja branżowa cytuje diagnozę, która sama w sobie jest opisem choroby: *„Anytime they saw something sexy it had to go into the OS”* — ilekroć zobaczyli coś atrakcyjnego, musiało trafić do systemu. Zamiast skupionej modernizacji jądra Copland zbierał niepowiązane ze sobą technologie: QuickDraw GX, SOM, OpenDoc. Każda z nich z osobna miała sens; razem dawały produkt, którego nie dało się ani przetestować, ani zintegrować. Wewnętrznie inżynierowie mówili wprost: *„There's no way in hell Copland ships next year. I just hope it ships in 1997.”*

Skala przedsięwzięcia: zespół liczący **około 500 osób** i budżet rzędu **250 milionów dolarów rocznie**. **Ellen Hancock**, zatrudniona jako dyrektor do spraw technologii między innymi po to, by ten projekt uratować, szybko doszła do wniosku, że Copland **nigdy nie zostanie wydany** — i to jej rekomendacja doprowadziła najpierw do zamrożenia prac, a potem do rozpoczęcia poszukiwań gotowego systemu na zewnątrz. PC World umieścił w 2008 roku Coplanda na liście największych porażek w historii branży.

> **O cytacie Amelio — i dlaczego go tu nie ma.** Krąży kilka wersji wypowiedzi Gila Amelio o Coplandzie. Najczęściej cytowana jego metafora — o statku z dziurą w dnie, który nabiera wody, i o tym, że jego zadaniem jest skierować ten statek we właściwą stronę — dotyczy **stanu firmy Apple**, a nie projektu Copland, i bywa do kontekstu anulowania projektu przypisywana błędnie. Osobno przypisuje się Amelio charakterystykę Coplanda jako zbioru oddzielnych projektów zamiast jednego systemu. Nie znalazłem w wiarygodnym źródle jednego kanonicznego cytatu Amelio o samym anulowaniu, więc świadomie żadnego nie podaję.

Dwa kolory fiszek: Pink i Taligent

Na sesji planistycznej Apple w **marcu 1988** pomysły na przyszłość systemu sortowano na fiszkach. **Niebieskie** dostawały usprawnienia możliwe do wdrożenia w krótkim horyzoncie. **Różowe** — koncepcje długoterminowe. **Czerwone** — rzeczy radykalne, na kiedyś.

Z fiszek niebieskich wyrósł **System 7**, a zespół, który go zbudował, nazywał się od koloru tych karteczek **„Blue Meanies”**. Z fiszek różowych wyrósł projekt **Pink**. Grupa inżynierów zwana „Gang of Five”, pod wodzą **Ericha Ringewalda**, zażądała dla Pinka autonomii albo odejścia z firmy — i autonomię dostała. Pink miał być obiektowym systemem z wywłaszczaniem, ochroną pamięci i zaawansowaną grafiką, czyli w gruncie rzeczy tym samym, co potem obiecywał Copland.

2 marca 1992 z Pinka powstała **Taligent Inc.** — osobna spółka Apple i IBM, ze **170 pracownikami** (w większości z Apple) i prezesem **Joem Guglielmim**. W styczniu 1994 **HP** dokupiło 15 procent udziałów. Stos technologiczny składał się z **TalOS** (obiektowy system na mikrokernelu), **TalAE** (framework aplikacyjny, przemianowany potem na **CommonPoint**) oraz **TalDS/TalDE** (narzędzia). Unicode, grafika trójwymiarowa, komponentowy model dokumentu — zestaw ambicji imponujący nawet dziś.

Diagnoza porażki mieści się w jednym zdaniu, które jest zarazem najlepszym cytatem w całym tym materiale: frameworki Taligenta były tak potężne, że *można było napisać dowolny program w trzech liniach kodu, ale zrozumienie, które to mają być trzy linie, zajmowało pół roku*. Doświadczonym programistom C++ dojście do pierwszej działającej aplikacji zajmowało **co najmniej trzy miesiące**.

Rozpad postępował szybko: **19 grudnia 1995** Apple i HP sprzedały udziały IBM-owi, a zwolnienia objęły blisko 200 z 375 osób; w **listopadzie 1996** porzucono natywny TalOS ostatecznie; **16 września 1997** IBM ogłosił rozwiązanie spółki; w **styczniu 1998** resztki wchłonął IBM.

A teraz rzecz, która czyni tę historię wartą opowiedzenia — bo z Taligenta przetrwało zaskakująco dużo:

- **Internacjonalizacja Javy. Mark Davis** kierował pracami, które dały obsługę Unicode w JDK 1.1–1.1.4, a potem dały początek otwartoźródłowemu **ICU** (International Components for Unicode) — bibliotece używanej dziś praktycznie wszędzie, od przeglądarek po bazy danych. Jeśli w dowolnym współczesnym systemie działa poprawne sortowanie tekstu w kilkudziesięciu językach, to jest w tym kawałek Taligenta.
- **Ludzie.** Erich Ringewald — ten sam, który zaczynał Pink — trafił do **Be Inc.**, gdzie projektował **BeOS**. Człowiek, który rozpoczął pierwszą próbę napisania następcy klasycznego Mac OS, skończył budując system, który Apple prawie kupiło osiem lat później. Rozmowy o tym zakupie opisuję w następnej sekcji.

Gershwin

Gershwin to nazwa robocza systemu, który miał być następcą Coplanda — dopełnieniem tego, czego Copland nie zdążył zrobić. Przede wszystkim chodziło o **pełną wielozadaniowość wywłaszczającą dla wszystkich aplikacji**, podczas gdy Copland miał ją dawać tylko nowemu

kodowi, a stary trzymać w kooperacyjnym „niebieskim polu”.

Gershwin nigdy nie wyszedł poza fazę planów i slajdów. Nie powstał żaden kod produkcyjny. Zniknął razem z Coplandem w 1996 roku. Bywa podawane, że miał planowaną datę wydania w 1997 albo 1998 roku, ale te daty pochodzą z prezentacyjnych map drogowych Apple, a nie z realnego harmonogramu prac — i tak też je tu traktuję.

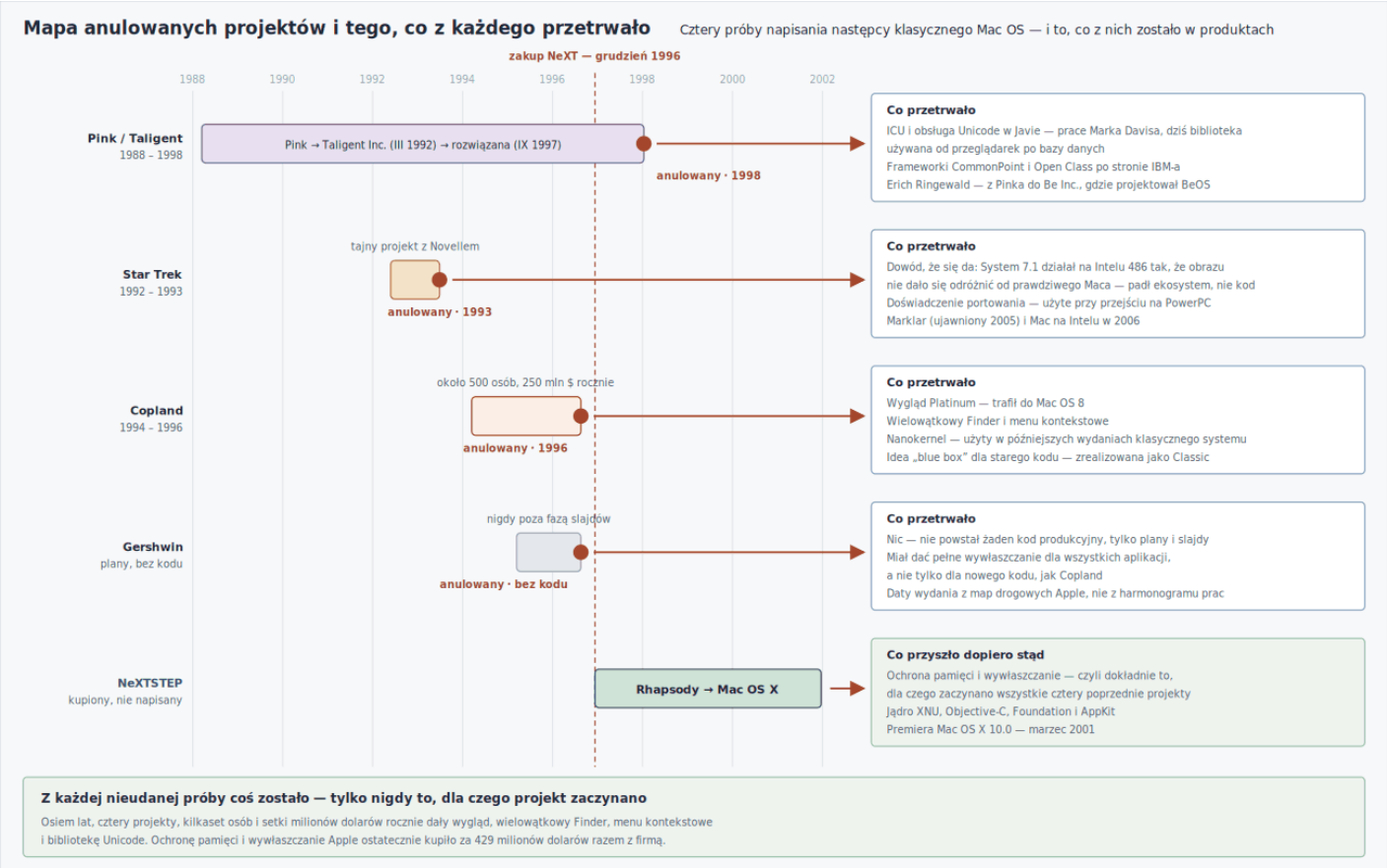
Star Trek: Mac OS na x86 (1992–1993)

Star Trek to tajny projekt z 1992 roku, mający uruchomić System 7 na komputerach PC z procesorami Intel x86. Apple dostarczało większość zasobów inżynierskich, a **Novell** zamierzał sprzedawać gotowy produkt jako konkurencję dla Windows. Nazwa wzięła się od hasła *to boldly go where no Mac has gone before*.

Osiągnięcie techniczne było realne i imponujące: zespołowi udało się uruchomić **System 7.1 na komputerze z procesorem Intel 486**, a obraz na ekranie był **nie do odróżnienia** od prawdziwego Maca. System miał działać na nadchodzącym DR DOS Novella. Fundamentalny problem leżał zupełnie gdzie indziej niż w kodzie: **każda aplikacja wymagałaby przeportowania na x86**. System działał, ale ekosystem nie — a bez ekosystemu produkt nie miał sensu rynkowego.

Dlaczego padł: zmiana władzy. John Sculley odszedł ze stanowiska dyrektora generalnego, zastąpił go **Michael Spindler**, który przekierował zasoby na migrację do PowerPC i uznał Star Trek za produkt niesprzedawalny. Doszły do tego walki wewnętrzne i problemy personalne.

Co przetrwało: komponenty wykorzystano przy przejściu Apple na PowerPC, a później przy portowaniu Mac OS X na procesory Intela — w projekcie **Marklar**, ujawnionym publicznie w 2005 roku. Natywny system Apple na Intelu trafił na rynek w **2006**, czternaście lat po Star Treku.



Rysunek 6. *Cztery nieudane próby napisania następcy nie poszły w całości do kosza: z każdej coś zostało w produktach, tylko nigdy to, dla czego projekt zaczynano — czyli ochrona pamięci i wywłaszczanie, które ostatecznie przyszły z kupionego systemu.*

Poszukiwanie nowego systemu, 1996–1997

Rozmowy z Be i kwota, której nie da się ustalić

Be Inc. założyli w 1990 roku **Jean-Louis Gassée**, były szef działu produktów Apple, i **Steve Sakoman**, wcześniej w Apple twórca i szef zespołu Newtona. BeOS był systemem nowoczesnym w każdym wymiarze, którego klasycznemu Mac OS brakowało: wywłaszczający, wielowątkowy, z ochroną pamięci, z obiektowym API w C++ i własnym systemem plików BFS. Działał na własnych maszynach Be — BeBoksach z dwoma procesorami PowerPC 603 — a od 1996 roku również na Macintoshach z PowerPC, w tym na klonach. Power Computing w listopadzie 1996 roku licencjonowało BeOS i dołączało go bez dopłaty do swoich klonów; spotykanej relacji, że pomagało firmie Be specyfikacjami wbrew stanowisku Apple, **nie udało się potwierdzić** (tekst **BeOS i**

Haiku). Udokumentowane jest za to, że pod koniec 1997 roku Apple oficjalnie odmówiło Be pomocy technicznej przy przeniesieniu BeOS na nowe Maki z procesorem G3.

> **Kwota żądana przez Gassée jest najbardziej sporną liczbą w całej tej historii.** Apple oferowało rząd **125 milionów dolarów**, a Gassée miał żądać — źródła podają — od **200** do **500 milionów**, w wersji skrajnej także wysokiego stanowiska w Apple[1]; wersję z pięciuset milionami traktuję jako niepotwierdzoną. Najlepiej udokumentowana, bo z prasy z epoki, jest para: Apple zaczęło od kwoty wyraźnie poniżej 100 milionów, a Gassée zszedł z „ponad 300” do 200 milionów. Do tego dochodzi zastrzeżenie, że **sam Gassée odrzucał krążące pogłoski jako nieścisłe**. Uczciwa formuła brzmi więc tak: Apple oferowało rząd stu dwudziestu pięciu milionów, Gassée oczekiwał wielokrotności tej sumy, a konkretna liczba nie jest ustalona i osoba najlepiej poinformowana zaprzeczała krążącym wersjom.

Warto jednak podkreślić, że rozmowy nie padły wyłącznie na kwocie i redukowanie ich do targu jest uproszczeniem. Powód merytoryczny był poważniejszy: **BeOS był niekompletny**. Krytykowano brak tak podstawowych rzeczy jak obsługa drukowania, brakowało dojrzałej obsługi sieci i szerokiego zestawu sterowników. Apple potrzebowało systemu, który da się dowieźć klientom w rok albo dwa; BeOS wymagałby dobudowania połowy warstwy usługowej, a to oznaczałoby powtórkę z Coplanda, tylko na cudzym kodzie. Do tego doszła — jak zgodnie podają źródła — siła perswazji Steve'a Jobsa.

Wybór NeXT

Przejęcie NeXT ogłoszono **20 grudnia 1996**, a sfinalizowano w **lutym 1997**. Jobs wrócił najpierw jako doradca Amelia; w lipcu 1997 Amelio odszedł, a we wrześniu Jobs został tymczasowym dyrektorem generalnym.

> **Kwota transakcji jest sporna, choć w węższym zakresie.** Źródła podają **400 milionów dolarów** albo **429 milionów dolarów w gotówce plus 1,5 miliona akcji Apple** przekazanych osobiście Jobsowi[2]. Przyjmuję 429 milionów plus 1,5 miliona akcji.

Skutek jest znany: NeXTSTEP i OPENSTEP stały się fundamentem systemu **Rhapsody**, a potem **Mac OS X**. Klasyczny Mac OS zszedł do roli środowiska zgodności, a jego API w okrojonej formie **Carbon** stało się mostem dla dużych producentów oprogramowania. Mechanikę samego NeXTSTEP-u — Mach, warstwę BSD, Objective-C, Interface Builder — opisuje tekst **NeXTSTEP i OPENSTEP**, a budowę i pierwsze lata następcy — **Mac OS X: przejście**.

Klony (1995–1997) i numer wersji jako instrument prawny

W 1995 roku Apple po raz pierwszy w historii zaczęło **licencjonować Mac OS** producentom zewnętrznym. Kontekst decyzji: Macintosh miał wtedy około **siedmiu procent** światowego rynku komputerów osobistych, a słabnące zainteresowanie IBM-a pecetami obniżało oczekiwania wobec sojuszu AIM. Licencjonowanie miało powiększyć rynek platformy.

Licencjobiorcy: **Power Computing**, **UMAX Technologies**, **Motorola Computer Group** z linią StarMax, **DayStar Digital** z wieloprocesorowymi maszynami Genesis MP, **Radius** i **APS Technology**. W sumie w erze licencjonowania wprowadzono na rynek **75 różnych modeli klonów**. Warunki: opłata licencyjna zryczałtowana plus tantiema, początkowo w wysokości **50 dolarów** od sprzedanego komputera. Dla Apple w kryzysie finansowym był to szybki przychód.

Czego Apple nie przewidziało: klony — a zwłaszcza maszyny Power Computing — okazały się **szybsze i tańsze** od komputerów Apple. Zamiast powiększać rynek, zjadały marżę Apple na sprzęcie. A Apple zarabiało na sprzęcie, nie na pięćdziesięciu dolarach tantiemy od sztuki. Mechanizm, który miał ratować platformę, zaczął dusić firmę, która ją utrzymywała.

Steve Jobs próbował w 1997 roku kilkakrotnie renegocjować umowy na korzyść Apple i — jego własnymi słowami — został *„basically told to pound sand”*. Odpowiedź, która potem nastąpiła, jest w tej serii bez precedensu.

> **Mac OS 8 był narzędziem prawnym, nie technicznym.** Licencje klonowe mówiły o „Systemie 7”. System wydany pod numerem **8** formalnie nie był objęty tymi umowami — był nowym produktem, którego nikt nie miał prawa dystrybuować. Klony zostały w ten sposób odcięte od aktualizacji, a przez to od rynku, bez zrywania jakiejkolwiek umowy i bez jednego procesu sądowego. Numer wersji systemu operacyjnego posłużył jako instrument prawny.

Finał był szybki. Apple **kupiło operacje Power Computing za 100 milionów dolarów** we wrześniu 1997 i rozdało ich użytkownikom darmowe aktualizacje do Mac OS 8 — co było zarazem gestem wobec klientów i likwidacją najgroźniejszego konkurenta. **UMAX jako jedyny uzyskał licencję na Mac OS 8**, wygasającą w **lipcu 1998**. Do końca 1997 roku praktycznie wszystkie pozostałe kontrakty klonowe zostały rozwiązane. Epizod licencjonowania trwał niecałe trzy lata i firma nie powtórzyła go nigdy.

Classic Environment: klasyczny system wewnątrz następcy

Classic to warstwa zgodności w Mac OS X pozwalająca uruchamiać aplikacje napisane na Mac OS 9. Działała jako warstwa abstrakcji sprzętu i oprogramowania, tworząca izolowaną piaskownicę dla starego kodu — i jest bezpośrednią realizacją idei „niebieskiego pola” z Coplanda, tyle że na jądrze Mach zamiast na mikrokernelu Apple.

Mechanizm był prosty w opisie i nieoczywisty w skutkach: do piaskownicy ładowano **pełną wersję Mac OS 9**, podmieniając część niskopoziomowych wywołań na odpowiedniki Mac OS X. Wymagało to zainstalowanej teczki systemowej Mac OS 9 — zalecano wersję 9.1 albo 9.2.2 — oraz pliku New World ROM. Na użytkownika przypadał **tylko jeden proces Classic**. Uruchamiać można było na trzy sposoby: przy starcie systemu, na żądanie z panelu preferencji albo automatycznie przy otwarciu pierwszej klasycznej aplikacji.

Kluczowa cecha architektoniczna tego rozwiązania jest zarazem ostatnim echem całego tego tekstu. Z punktu widzenia jądra Mach **cały Mac OS 9 wraz ze wszystkimi uruchomionymi w nim aplikacjami był jednym procesem** Mac OS X. Oznaczało to, że aplikacje klasyczne były chronione przed resztą systemu — ale **nadal nie były chronione przed sobą nawzajem**. Awaria jednej klasycznej aplikacji mogła położyć całe środowisko Classic. Nie mogła jednak tknąć Mac OS X.

Zakres wsparcia to jedna z tych dat, które warto pamiętać: **Classic działało wyłącznie na Macach z procesorem PowerPC, w wersjach Mac OS X do 10.4 Tiger włącznie**. Nie działało w 10.5 Leopard i **nigdy nie działało na Macach z procesorami Intela**. Konsekwencja była ostra: dla kogoś, kto kupił Maca z Intelem w 2006 roku, klasyczne oprogramowanie przestało być uruchamialne w ogóle. Zostały emulatory albo stary sprzęt. Jest to cezura znacznie twardsza niż stopniowe wygaszanie Classic na PowerPC — i warto ją znać, bo dotyczy każdego, kto dziś ma w archiwum pliki z tamtej epoki.

Największe wady

Brak ochrony pamięci

Jedna źle napisana aplikacja albo jedno rozszerzenie kładło całą maszynę i nie było sposobu, żeby się przed tym zabezpieczyć administracyjnie. Nie chodzi o to, że nie było dobrych praktyk — chodzi o to, że **nie istniał żaden mechanizm, który mógłby je wyegzekwować**. Administrator mógł standaryzować obraz, ograniczać listę dozwolonych programów i uczyć ludzi, ale nie mógł postawić granicy, której program nie przekroczy, bo takiej granicy w systemie nie było.

Brak wywłaszczania

Zapętlony program blokował wszystko: interfejs, inne aplikacje, pracę w tle. Nie dało się zabić procesu w sposób czysty. Command-Option-Escape był loterią, po której i tak zalecano restart — i to zalecenie pochodziło od producenta systemu, nie od zniechęconego użytkownika.

Ręczne zarządzanie pamięcią przez użytkownika

Przydziałów w oknie „Informacje” trzeba było uczyć każdego użytkownika, łącznie z tym, który przyszedł do pracy składać teksty, a nie administrować systemem. Komunikat „za mało pamięci” przy czterdziestu wolnych megabajtach nie daje się wytłumaczyć bez wykładu o partycjach aplikacji — a wykład o partycjach aplikacji nie jest tym, czego oczekuje się od systemu reklamowanego jako prosty.

Konflikty rozszerzeń

Dwie identyczne maszyny mogły zachowywać się inaczej, a diagnostyka zajmowała godziny, przy czym znaczna jej część polegała na restartowaniu komputera i patrzeniu, czy tym razem będzie lepiej. Nie było mechanizmu izolacji, nie było podpisu, nie było kontroli wersji ani rejestru zmian w teczce systemowej. Narzędzia takie jak Conflict Catcher były znakomite — i były łataniem dziury, której system nie powinien był mieć.

Brak wielodostępności

Multiple Users w Mac OS 9 z 1999 roku było kosmetyką na poziomie powłoki: bez izolacji systemu plików, bez izolacji pamięci, bez uprawnień. Przez piętnaście lat Mac był systemem jednoosobowym, co w pracowni komputerowej albo w studiu na dwie zmiany oznaczało, że wszyscy pracują na jednym koncie i wszyscy widzą wszystko.

Brak modelu bezpieczeństwa

Nie było uprawnień plikowych. Nie było kontroli tego, co może zrobić uruchomiony program. Hasła w File Sharing były słabe, a sama usługa — włączana chętnie i szeroko, bo działała bez serwera. Model bezpieczeństwa sprowadzał się do założenia, że w sieci AppleTalk w biurze nie ma nikogo złośliwego, co w 1991 roku było założeniem prawie rozsądnym, a w 1999 już nie.

Marnotrawstwo miejsca w HFS

Bloki alokacji po szesnaście czy trzydzieści dwa kilobajty na dużych dyskach oznaczały realną utratę znacznej części pojemności na woluminach z wieloma małymi plikami — skali procentowej nie podaję, bo żadne ze źródeł jej nie kwantyfikuje. Obejściem było partycjonowanie dysku wyłącznie po to, żeby zmniejszyć blok — czyli administracyjna praca wykonywana po to, by skompensować szesnastobitowe pole w strukturze danych z 1985 roku.

Widelce zasobów jako bariera interoperacyjności

Każde przejście pliku przez świat nie-macowy groziło jego cichym uszkodzeniem. MacBinary, BinHex i pliki `._` to trzy pokolenia obejść tego samego problemu, a żadne z nich nie było przezroczyste. Dla administratora sieci mieszanej był to stały koszt.

Brak wiersza poleceń

AppleScript był dobry i wyprzedzał epokę — ale to nie jest powłoka. Nie dało się napisać skryptu, który przejdzie po katalogu, przefiltruje pliki i wykona coś na każdym, w sposób, w jaki robi się to jedną linią w każdym systemie uniksowym. Zdalne zarządzanie flotą wymagało osobnych, drogich produktów w rodzaju Apple Network Assistant. Dla administratora jest to, obok braku uprawnień, najbardziej uciążliwa cecha tego systemu — i pierwsza, której zniknięcie odczuł po przejściu na Mac OS X.

Zamrożenie architektury na siedemnaście lat

Między 1984 a 2001 rokiem fundamenty systemu nie zmieniły się mimo czterech prób ich wymiany. Każda kolejna wersja dokładała funkcje na tej samej podstawie, a podstawa starzała się coraz szybciej, bo konkurencja — opisana w tekstach **Windows 95**, **OS/2** i **BSD** — miała ochronę pamięci i wywłaszczanie znacznie wcześniej. Nowy fundament przyszedł dopiero z zakupem cudzej firmy.

> **Uczciwa druga strona, bez której ta lista byłaby nieprawdziwa.** Przez większość opisywanego okresu klasyczny Mac OS był **łatwiejszy w obsłudze, spójniejszy i po prostu przyjemniejszy w pracy** od wszystkiego dookoła. Konfiguracja sprzętu była bezbolesna — żadnych przerwań IRQ do ustawiania ręcznie, urządzenia na SCSI i ADB

> rozpoznawane same. Czcionki i kolor działały lepiej niż gdziekolwiek indziej, co zdecydowało o całej branży DTP. A model „jeden plik to jedna aplikacja” do dziś nie doczekał się lepszego odpowiednika: instalacja polegała na przeciągnięciu ikony, a deinstalacja na wyrzuceniu jej do kosza. Ludzie nie zostawali przy tym systemie z przyzwyczajenia.

Praktyka dziś: emulatory i legalne źródła

Emulatory

- **Mini vMac** — emuluje bardzo wczesnego Maca, domyślnie Macintosha Plus z Systemem 6. Najlżejszy i najwierniejszy dla epoki 1984–1990. To jest narzędzie, którym pokazuje się MFS, wieczny pusty folder i tryb wymiany dysków.
- **Basilisk II** — maszyny 68k klasy Macintosh II i Quadra, systemy od Systemu 7 do Mac OS 8.1.
- **SheepShaver** — Mac z procesorem PowerPC na poziomie G4, systemy **do Mac OS 9.0.4** włącznie.
- **QEMU** — emuluje maszyny PowerPC Mac poleceniem `qemu-system-ppc` z maszyną `mac99`, co pozwala uruchamiać Mac OS 9.2.x oraz wczesne wersje Mac OS X. Wymaga OpenBIOS albo obrazu ROM, zależnie od konfiguracji.
- **Infinite Mac** — Mini vMac, Basilisk II i SheepShaver skompilowane do WebAssembly i działające w przeglądarce, z gotowymi obrazami dysków. Najszybszy sposób, żeby komuś pokazać System 7 — otwiera się kartę i już. Status prawny dołączonych obrazów warto sprawdzić przed linkowaniem tego w materiale firmowym.
- **PearPC** — historyczny, nierozwijany; wymieniam dla porządku.

Mac OS 9.0.4 w emulatorze SheepShaver — najwyższa wersja, jaką ten emulator obsługuje. Zwróć u

Mac OS 9.0.4 w emulatorze SheepShaver — najwyższa wersja, jaką ten emulator obsługuje. Zwróć uwagę, że to wciąż ten sam system bez ochrony pamięci, tylko z nowszym wyglądem i funkcjami wydania 9. Zrzut ekranu: Apple; przytoczony w celu omówienia (prawo cytatu, art. 29 pr. aut.). Plik: en.wikipedia, non-free.

> **Kwestia legalności.** Obrazy ROM Macintoshy są objęte prawem autorskim Apple i **nigdy nie zostały oficjalnie udostępnione**. Jedyna bezpieczna ścieżka to zrzucenie ROM-u z fizycznie posiadanej maszyny. Archiwa społecznościowe — **Macintosh Garden**, **Macintosh Repository**, **WinWorld** — mają status prawny w najlepszym razie szary; są to archiwa oprogramowania porzuconego, co nie jest kategorią prawną, tylko obyczajową. W materiale

> firmowym należy to zaznaczyć, a nie przemilczeć.

Co udostępniło Apple

Apple przez lata udostępniało w sekcji „Older Software Downloads” obrazy starszych systemów, między innymi **System 7.5.3** wraz z aktualizacją **7.5.5**. Strony te były wielokrotnie reorganizowane i część z nich zniknęła, więc przed odesłaniem kogokolwiek pod konkretny adres trzeba sprawdzić aktualny stan witryny wsparcia Apple — nie podaję tu adresu, bo ryzyko, że będzie martwy, jest wysokie.

Apple udostępniło też narzędzie **MFSLives**, dające dostęp tylko do odczytu do woluminów MFS. Dla kogoś, kto ma w archiwum dyskietki z 1985 roku, jest to ścieżka odzysku danych pochodząca wprost od producenta — rzecz warta odnotowania, bo producenci rzadko dbają o formaty, które sami wycofali.

Ściągawka

Zagadnienie	Jak to było w klasycznym Mac OS
Pierwsze i ostatnie wydanie	System 1.0 (I 1984) — Mac OS 9.2.2 (5 XII 2001)
Gdzie leżał system	Toolbox w 64 KB ROM na płycie; plik `System` jako zbiór zasobów i łatek
Ochrona pamięci	Brak przez całe 17 lat; przyszła dopiero z jądrem Mach w Mac OS X
Wielozadaniowość	Kooperacyjna od MultiFindera (1987); oddanie procesora przez `WaitNextEvent()`
Model pamięci	Bloki relokowalne adresowane uchwytami, blokada przez `HLock`; partycje z zasobu `SIZE`
Kto przydzielał pamięć aplikacji	Użytkownik, ręcznie, w oknie Get Info
Systemy plików	MFS (płaski, do 20 MB) → HFS (1985, B-drzewo) → HFS+ (Mac OS 8.1, 1998)
Struktura pliku	Widelec danych + widelec zasobów; typ i twórca zamiast rozszerzenia nazwy
Rozszerzanie systemu	Łatanie pułapek przez `NGetTrapAddress`/`NSetTrapAddress` — technika oficjalna
Diagnostyka	Start z klawiszem Shift, Extensions Manager (spacja), Conflict Catcher
Awaria	Bomba z numerem błędu; Command-Option-Escape i tak kończył się restartem

Zagadnienie	Jak to było w klasycznym Mac OS
Architektury procesora	68000 → 68040 → PowerPC (1994); emulator 68k obecny do końca linii
Wielu użytkowników	Dopiero Mac OS 9 (1999), i tylko na poziomie powłoki
Wiersz poleceń	Nie było; AppleScript od 1993 jako namiastka automatyzacji
Koniec	Classic w Mac OS X do 10.4 na PowerPC; na Intelu nigdy

Mity i sprostowania

„Cały system mieścił się w 400 kilobajtach”

Nie mieścił się. Połowa systemu — cały Toolbox — siedziała w 64 kilobajtach pamięci stałej na płycie głównej, a plik na dyskietce był nakładką i zbiorem łatek. Pytanie „jak to się zmieściło” zakłada nieprawdziwy stan rzeczy.

„System 1 miał foldery”

Miał ich wizerunek. MFS był płaski, a przynależność do folderu była polem w rekordzie pliku. Dowód: foldery były widoczne w Finderze i niewidoczne w oknach Otwórz i Zapisz, które pokazywały płaską listę całego woluminu.

„System 7 dodał ochronę pamięci”

Dodał pamięć wirtualną, czyli stronicowanie na dysk. Wszystko nadal siedziało w jednej przestrzeni adresowej. Więcej pamięci, nie bezpieczniejsza.

„AppleScript przyszedł z System 7.0”

System 7.0 przyniósł Apple Events, czyli warstwę, na której AppleScript stanął. Sam język pojawił się w 1993 roku z System 7.1.1, a powszechnie dostępny stał się od System 7.5.

„Rozszerzenia były hackiem”

Łatanie pułapek było udokumentowaną techniką Apple, z oficjalnym API, używaną przez sam system do poprawiania kodu w ROM-ie. Problemem nie był mechanizm, tylko to, że nikt nie kontrolował, kto z niego korzysta i w jakiej kolejności.

Epilog

Klasyczny Mac OS umarł dwa razy. Raz technicznie — w sierpniu 1996, kiedy Ellen Hancock zamroziła Coplanda i stało się jasne, że Apple nie napisze następcy samo. Raz ceremonialnie — na scenie WWDC w 2002 roku, z trumną i muzyką organową. Między jedną a drugą datą minęło sześć lat, w trakcie których system dostał trzy z najlepszych wydań w swojej historii: Mac OS 8, 8.5 i 9. Jest w tym pewna nauka o tym, jak wygląda dobrze przeprowadzone wygaszanie: kiedy firma przestała udawać, że klasyczny system ma przyszłość, zaczęła go wreszcie porządnie rozwijać jako to, czym był — narzędziem, którego ludzie używają dziś.

Teza tego tekstu brzmiała, że wszystkie zalety i wszystkie wady tego systemu mają jedno źródło, i chciałbym ją na koniec doprecyzować. Nie chodzi o to, że brak ochrony pamięci był ceną za otwartość, bo to byłby handel, którego nikt świadomie nie zawarł. Chodzi o coś prostszego i mniej wygodnego: **w 1983 roku nie dało się tego zaprojektować inaczej**. Sto dwadzieścia osiem kilobajtów pamięci, procesor bez jednostki zarządzania pamięcią, brak dysku twardego — przy tych parametrach izolacja procesów nie była kompromisem odrzuconym, tylko możliwością nieistniejącą. Decyzje z 1983 roku były poprawne.

Błąd — jeśli szukać w tej historii błędu — polega na czymś innym. Polega na tym, że przez trzynaście lat, od 1983 do 1996, żadna z prób wymiany fundamentu nie została doprowadzona do końca, a każda kolejna była ambitniejsza od poprzedniej. Pink miał być obiektowym systemem przyszłości. Copland miał mieć mikrokernel, ochronę pamięci, wywłaszczanie, OpenDoc, SOM i QuickDraw GX naraz. Gershwin miał naprawić to, czego Copland nie naprawiał. Za każdym razem zakres rósł szybciej niż zespół był w stanie go dowieźć — i za każdym razem produktem końcowym była lista technologii przeniesionych do starego systemu.

> **To jest, moim zdaniem, najbardziej praktyczny wniosek z całej tej historii.** System, którego fundament wymaga wymiany, rzadko ginie przez ten fundament. Ginie przez to, że każda kolejna próba jego wymiany jest większa od poprzedniej, bo trzeba dogonić także wszystko, co w międzyczasie dołożono. Apple wyszło z tej pętli jedynym wyjściem, jakie jej zostało: kupiło gotowy system od kogoś, kto miał go skończony od siedmiu lat. Kosztowało to 429 milionów dolarów i oddanie firmy jej byłemu założycielowi — i było tańsze niż piąta próba napisania Coplanda.

Ciekawostki zebrane

- **Dźwięk awarii Power Macintoshy 6100, 7100 i 8100 to zmontowany dźwięk wypadku samochodowego**, złożony z próbek z bibliotek Elektra Records i Hanna-Barbera. Maszyna, która nie wstawała, informowała o tym odgłosem stłuczki.
- Kultowy startowy „bong”, znany z Quadry 840AV i potem z całych pokoleń Maców, wprowadził **Jim Reekes po cichu, bez zgody przełożonych**. Nie znosił poprzedniego dźwięku, bo kojarzył mu się trytonem z dźwiękami awarii. Akord Fis-dur wywodzący się z tego nagrania został w 2012 roku zastrzeżony w USA jako znak towarowy, a w 2008 zagrał w filmie „WALL-E”.
- Pozostałe dźwięki awarii — *Chimes of Death* — różniły się zależnie od maszyny: Macintosh II miał wznoszące arpeggio durowe, Quadra i Centris jego niższy wariant, Quadra/Centris 660AV sampel z syntezatora Roland D-50, a Power Maki PCI sprzed serii G3 — pękanie i tłuczone szkło, już bez ikony Sad Mac.
- Startowy „beep” Macintosha 128K to fala prostokątna 600 Hz, zaprogramowana przez **Andy'ego Hertzfelda** na układzie MOS 6522 VIA. Akord pierwszych Power Macintoshy zagrał **Stanley Jordan** na gitarze Ovation, techniką tappingu.
- **Happy Mac**, uśmiechnięta ikonka poprawnego startu, też wyszła spod ręki Susan Kare; jako inspiracje podaje się obudowę kompaktowego Macintosha i postać Two-Face'a z Batmana.
- **Sosumi** — nazwa systemowego dźwięku alarmowego to fonetyczny zapis frazy „so sue me”. Żarcik wymierzono w Apple Records, procesujące się z Apple Computer o nazwę w kontekście muzycznym: dźwięki musiały mieć nazwy niemuzyczne, więc nazwano ten tak, żeby brzmiał niewinnie i znaczył dokładnie to, co znaczył.
- **„Stolen from Apple Computer”** — ikona ukryta w ROM-ie wczesnych Maców, przygotowana jako dowód skopiowania pamięci stałej na wypadek procesu przeciw producentom klonów sprzętowych.
- System był pełen **easter eggów**: wpisanie frazy „secret about box” w Notatniku, zaznaczenie jej i przeciągnięcie na biurko odsłaniało planszę z animowaną flagą i listą twórców; przytrzymanie klawisza Option przy „About This Macintosh” pokazywało oryginalną grafikę z 1984 roku; a w ResEdit kombinacja Option-Shift-Command włączała „tryb świni”.
- **Chicago**, systemowa czcionka interfejsu do Mac OS 7.6, też jest autorstwa Susan Kare. Apple nazywało czcionki nazwami miast; od Mac OS 8 systemową czcionką została **Charcoal**.
- Autor Conflict Catchera, najlepszego narzędzia do rozplątywania konfliktów rozszerzeń, **Jeffrey Robbin**, napisał później SoundJam MP — program, z którego Apple zrobiło **iTunes** .
- **Bomba systemowa** — ikona bomby z zapalonym lontem narysowana przez **Susan Kare**, komunikat „Sorry, a system error occurred”, numer błędu i przycisk „Restart” obok zwykle nieklikalnego „Resume”. Numery to wyjątki procesora: **01** to bus error, **02** address error (dostęp pod adres nieparzysty, klasyk na 68000), **03** illegal instruction, **11** wyjątek sprzętowy, **25** brak pamięci, **26** nie da się uruchomić aplikacji, **28** przepełnienie stosu.
- **Sad Mac** — ikona wyświetlana przy starcie, gdy testy sprzętowe wykryły problem uniemożliwiający uruchomienie, z kodem szesnastkowym w dwóch częściach: klasa błędu i podklasa. Nigdy nie została zaktualizowana do koloru, a na Power Makach z magistralą PCI zniknęła w ogóle — awarię startu sygnalizował tam już sam dźwięk, a od iMaca z 1998

roku seria tonów wskazujących typ błędu.

- **Podpisy ekipy w obudowie.** Wnętrze obudowy Macintosha 128K nosiło odlane podpisy zespołu projektowego — między innymi Jobsa, Burrella Smitha, Hertzfelda, Kare i Atkinsona. Jobs argumentował, że artyści podpisują swoje dzieła. Były niewidoczne dla użytkownika, bo obudowy nie wolno było otwierać bez specjalnego, długiego klucza Torx. Najczęściej podaje się 47 nazwisk, ale ta liczba bywa różna i jej nie potwierdzam.
- **Dźwięk startowy zmieniał się razem z maszynami.** Macintosh II witał użytkownika akordem C-dur autorstwa **Marka Lentcznera**, Quadry 700–800 — akordem C-dur **Jima Reekesa** zagranym na Korgu Wavestation EX, a modele 5200–6300 — kompozycją zrobioną na Fairlighcie CMI.

Oś czasu

Data	Wydarzenie
24 I 1984	Premiera Macintosha 128K z Systemem 1.0; MFS, Toolbox w 64 KB ROM
IV 1985	System 2.0: polecenia New Folder i Shut Down, LaserWriter, AppleTalk
IX 1985	HFS debiutuje razem z dyskiem Hard Disk 20
I 1986	System 3.0 z Macintoshem Plus: HFS wbudowany w ROM
X 1987	System Software 5.0 wprowadza MultiFinder i wielozadaniowość kooperacyjną
III 1988	Sesja planistyczna z fiszkami: niebieskie dadzą System 7, różowe projekt Pink
10 VI 1988	System 6.0
13 V 1991	System 7.0 — największy skok w historii linii; tego samego dnia System 6.0.8
2 X 1991	Zawiązanie sojuszu AIM: Apple, IBM, Motorola
14 II 1992	Start tajnego projektu Star Trek: System 7 na x86
2 III 1992	Powstaje Taligent Inc. jako spółka Apple i IBM
VIII 1992	System 7.1: folder Fonts, WorldScript, System Enablers
połowa 1993	Anulowanie projektu Star Trek
III 1994	Start prac nad Coplandem
14 III 1994	Pierwsze Power Macintoshe (6100/7100/8100) z System 7.1.2
IX 1994	System 7.5
1995	Apple zaczyna licencjonować Mac OS producentom klonów

Data	Wydarzenie
III 1995	System 7.5.1 — pierwszy ekran startowy z logo „Mac OS”
28 VII 1995	Premiera CommonPoint 1.1 — jedyny produkt Taligenta
19 XII 1995	Apple i HP sprzedają udziały w Taligencie IBM-owi
VIII 1996	Developer Release 0 Coplanda okazuje się bezużyteczny; Ellen Hancock zamraża projekt
20 XII 1996	Apple ogłasza przejęcie NeXT
I 1997	Mac OS 7.6 — pierwsza wersja oficjalnie nazwana „Mac OS”
26 VII 1997	Mac OS 8; licencje klonowe obejmowały tylko „System 7”
16 IX 1997	IBM ogłasza rozwiązanie spółki Taligent
IX 1997	Apple kupuje operacje Power Computing za 100 mln USD
I 1998	Mac OS 8.1 wprowadza HFS+; ostatnia wersja na procesory 68k
VII 1998	Wygasa licencja UMAX — koniec ery klonów
X 1998	Mac OS 8.5 — pierwsze wydanie wyłącznie dla PowerPC
V 1999	Mac OS 8.6: nanokernel z Multiprocessing Services 2.x
23 X 1999	Mac OS 9
5 XII 2001	Mac OS 9.2.2 — ostatnia wersja klasycznego Mac OS
2002	Inscenizowany pogrzeb Mac OS 9 na konferencji WWDC
2007	Koniec środowiska Classic wraz z końcem wsparcia Mac OS X 10.4 na PowerPC

Wątek polski

> **Lokalizacja przez podmianę zasobów w ResEdit była oficjalnym modelem Apple, a nie obejściem.** To jedno zdanie tłumaczy zjawisko, którego inaczej wytłumaczyć się nie da: dlaczego w kraju, dla którego Apple nigdy nie wydało pełnej polskiej wersji systemu, powstał oddolny ekosystem polonizatorów. Nie trzeba było łamać zabezpieczeń ani odtwarzać kodu — wystarczyło otworzyć plik w darmowym narzędziu producenta, podmienić zasoby `STR#`, `MENU`, `DLOG` i `DITL`, poszerzyć pola dialogów, bo polskie napisy są dłuższe od angielskich, i zapisać. Producent systemu sam zalecał ten sposób internacjonalizacji. Polscy tłumacze robili dokładnie to, co robiłby oddział Apple w kraju, w którym taki oddział by działał.

Apple nigdy nie wydało pełnej polskiej wersji klasycznego Mac OS na wzór niemieckiej czy francuskiej — tego nie udało mi się podważyć, choć oznaczam to jako ustalenie oparte na braku

dowodu przeciwnego, nie na dokumencie. Obsługa polskich znaków była natomiast możliwa i szła trzema kanałami:

- **Układ klawiatury** — zasoby `KCHR` mapujące klawisze na znaki. Polskie litery uzyskiwano układem „programisty” z klawiszem Option: Option i A dawało ą, Option i E dawało ę.
- **Kodowanie** — standardowe Mac OS Roman nie zawiera kompletu polskich znaków, więc istniało osobne **Mac OS Central European**, obsługiwane od System 7.1 przez **WorldScript**.
- **Czcionki** — polskie wersje krojów Type 1 i TrueType z kompletem diakrytyków przygotowywały polskie studia. Był to realny rynek, bo bez nich DTP po polsku po prostu nie działało.

Z kodowaniem wiąże się osobny, uciążliwy rozdział polskiej informatyki tamtych lat: **wymiana plików tekstowych z pecetem wymagała konwersji między Mac OS Central European, ISO Latin-2, Windows-1250 i Mazovią**. Cztery kodowania tego samego alfabetu, żadne domyślnie zgodne z pozostałymi, a przy tym każdy kanał wymiany — dyskietka, modem, później poczta — miał własne pułapki. Kto pracował wtedy między Makiem a pecetem, robił konwersję kodowania kilka razy dziennie i traktował to jak mycie rąk.

Co do obecności sprzętu: potwierdzona jest rola Macintosha w **polskiej poligrafii i DTP pierwszej połowy lat dziewięćdziesiątych**. Relacja Leopolda Tupalskiego mówi, że komputery z tej rodziny spotykało się wtedy w firmach zajmujących się usługami poligraficznymi lub DTP; potocznie nazywano je **„makówkami”**. Ta sama relacja potwierdza obecność Maców w szkolnictwie — pracownię w Liceum Ogólnokształcącym w Strzelnie wyposażoną w kilka Macintoshy LC i jedną Quadrę. Jest to jeden z nielicznych konkretów, jakie udało mi się znaleźć.

Dlaczego akurat DTP, a nie biuro? Wyjaśnienie jest dobrze umocowane technicznie, ale pozostaje moją hipotezą, a nie ustaleniem źródłowym: zestaw PageMaker albo QuarkXPress plus Photoshop plus Illustrator plus naświetlarka PostScript plus czcionki Type 1 istniał w dojrzałej formie wyłącznie na Macintoshu. Adobe i Aldus rozwijały swoje produkty najpierw na Maca, doszło do tego zarządzanie kolorem przez ColorSync od 1993 roku i natywna obsługa czcionek PostScript. Windows 3.1 nie był w tym segmencie konkurencyjny aż do połowy dekady — więcej o tamtym rynku jest w tekście **Windows 95**.

Ślad polonizatorów jest realny: pakiety zestawów zasobów podmieniających napisy interfejsu i dodających układy klawiatury krążą w obiegu retro do dziś, znalazłem ślad takiego pakietu przygotowanego dla Mac OS 9.

Źródła i dalsze lektury

Przypisy

[1] Pierwsza oferta Apple „well south of $100 million”, żądanie Gassée „300 mln plus”, zredukowane do połowy grudnia 1996 do 200 mln: „Business Week”, „The Bee In Be’s Bonnet” (9 II 1997); 300 mln wobec oferty 125 mln: Wikipedia — BeOS (za Low End Mac); ponad 300 mln: Low End Mac (2013); 200 mln wobec oferty nie wyższej niż 125 mln: 512 Pixels (2012); 275 mln: OSnews (2011), relacja z wypowiedzi Gassée; 500 mln plus wysokie stanowisko — pogłoska, którą sam Gassée odrzucał jako nieścisłą: Wikipedia — Be Inc.; poza tym spotyka się ją w popularnych tekstach, ale nie potwierdza jej żadne źródło odnalezione przy pracy nad tekstem **BeOS i Haiku**. Wątek stanowiska ma najbliższy odpowiednik w felietonie Stewarta Alsopa, według którego Gassée żądał kontroli nad rozwojem oprogramowania systemowego Apple — to relacja jednego autora: „Fortune” (3 II 1997).

[2] 400 mln USD: Wikipedia — Be Inc., także Stewart Alsop, „Fortune” (3 II 1997); 429 mln USD w gotówce plus 1,5 mln akcji Apple — częściej w literaturze i w komunikacie prasowym Apple. Rozbieżność bierze się stąd, że część źródeł podaje kwotę zaokrągloną albo wyłącznie część gotówkową.

Lektury

- Wikipedia (wersja angielska): hasła *Classic Mac OS*, *Macintosh File System*, *Hierarchical File System (Apple)*, *System 6*, *System 7*, *Mac OS 8*, *Mac OS 9*, *Mac 68k emulator*, *Resource fork*, *Extension conflict*, *Conflict Catcher*, *Classic (Mac OS X)*, *Copland*, *Taligent*, *Star Trek project*, *Be Inc.*, *Macintosh clone*, *MODE32*, *Sad Mac*, *Gil Amelio*.
- Apple Developer Library (archiwum): *Introduction to Processes and Tasks*, *Trap Manager*, *Introduction to Memory Management*.
- HandWiki: *Mac OS memory management*.
- Low End Mac: *System 7.5 and Mac OS 7.6: The Beginning and End of an Era*, *Apple's Copland Project*, *1997: Beleaguered, Mac OS 7.6 and 8, Killing Clones*.
- E-Maculation — przewodniki konfiguracyjne do Mini vMac, Basilisk II i SheepShaver.
- eeggs.com — archiwum easter eggów, w tym zestaw dla Mac OS 7.5.
- tupalski.eu — relacja Leopolda Tupalskiego o Macintoshach w Polsce lat dziewięćdziesiątych.

Ilustracje

Diagramy są moje. Zdjęcia i zrzuty ekranu na wolnych licencjach pochodzą z Wikimedia Commons. Zrzuty ekranu systemów komercyjnych pochodzą z angielskiej Wikipedii, gdzie są przechowywane jako pliki non-free w obniżonej rozdzielczości. Zrzuty wydanych wersji produktów Microsoftu wykorzystujemy na podstawie zasad Microsoftu dotyczących zrzutów ekranu, które wymagają dopisku „Used with permission from Microsoft”. Pozostałe zrzuty non-free — w tym systemów innych producentów, wersji testowych, ekranów startowych i zrzutów zawierających cudze treści —

przytaczamy w celu omówienia, na zasadzie prawa cytatu (art. 29 ustawy o prawie autorskim i prawach pokrewnych). Przy każdym zrzucie podstawa i licencja stoją w podpisie.

Nota metodyczna: czego nie wiemy

- **Rzeczywisty rozmiar pliku `System` w 1984 roku.** Krążące liczby (najczęściej 216 KB) nie mają potwierdzenia w źródle pierwotnym. Wartość pewną dałoby zważenie obrazu dyskietki w emulatorze; nie wykonałem tego.
- **Liczba wymian dyskietki przy kopiowaniu na Macintoshu 128K.** Powtarzane pięćdziesiąt do sześćdziesięciu jest zgodne z rachunkiem, ale nie znalazłem źródła Apple.
- **Jaki procent systemu pozostawał kodem 68k w kolejnych wydaniach.** Wszystkie krążące liczby to szacunki prasy i deweloperów; Apple nigdy nie podało danych. Dlatego w tekście nie ma żadnego procentu, są wyłącznie fakty jakościowe (Finder do Mac OS 8, emulator obecny do 9.2.2).
- **Intencja Motoroli wobec pułapek A-line i skala strat pojemności w HFS.** Sam mechanizm łatania pułapek oraz zjawisko zbyt dużych bloków alokacji są udokumentowane; twierdzenie, że producent procesora zarezerwował instrukcje A-line właśnie po to, by dało się rozszerzać system, oraz procentowa skala marnotrawstwa miejsca — nie.
- **Kwota żądana przez Jeana-Louisa Gassée za Be Inc.** Warianty rozciągają się od 200 do 500 milionów dolarów[1], sam Gassée dementował krążące pogłoski, a wersja kanoniczna nie istnieje.
- **Dokładna kwota przejęcia NeXT.** Podaję 429 milionów w gotówce plus 1,5 miliona akcji, odnotowując wariant „około 400 milionów”[2]; nie zweryfikowałem tego w dokumencie transakcyjnym.
- **Cytat Gila Amelio o anulowaniu Coplanda.** Kanonicznego cytatu nie ma. Najczęściej przypisywana mu metafora o statku z dziurą w dnie dotyczy stanu firmy, nie projektu — dlatego nie cytuję jej w kontekście Coplandu.
- **Planowana data wydania Gershwina.** Podawane lata 1997 i 1998 pochodzą z map drogowych prezentacyjnych, nie z harmonogramu prac.
- **Brak pełnej polskiej wersji klasycznego Mac OS.** Opieram to na braku jakiegokolwiek śladu takiego wydania, a nie na oświadczeniu Apple — jest to argument z milczenia i tak go traktuję.

Oceny podane w tym tekście jako oceny — w szczególności teza spinająca całość, że możliwość głębokiej przeróbki systemu i jego podatność na awarie mają jedno wspólne źródło; interpretacja modelu ręcznego przydziału pamięci jako przerzucenia na człowieka zadań jądra; wniosek z paradoksu Mac OS 8.6, że wywłaszczanie na dole stosu nie daje wywłaszczania na górze; ocena, że brak powłoki i brak uprawnień były najdotkliwszymi wadami tego systemu z punktu widzenia administratora; teza epilogu, że klasyczny Mac OS zginął nie przez swój fundament, lecz przez rosnący zakres kolejnych prób jego wymiany; oraz hipoteza tłumacząca dominację Macintosha w polskim DTP składem oprogramowania graficznego dostępnego wyłącznie na tej platformie — są

moją syntezą zebranego materiału, a nie ustaleniem, które dałoby się zacytować za źródłem.

Revision #6
Created 2026-09-27 08:08:15 UTC by Tomasz Konieczny
Updated 2026-09-28 19:56:17 UTC by Tomasz Konieczny