Specyfikacja systemu plików MBFS

Koncepcja do Laboratorium 6
z przedmiotu Systemy Operacyjne

Typ VULONG

Typ reprezentujący liczbę całkowitą z zakresu [0, 2^{63} - 1] zapisywany na zmiennej liczbie bajtów (od 1 do 9) w zależności od wartości. Każdy bajt kodujący liczbę oprócz ostatniego ma ustawiony najbardziej znaczący bit. Pozostałe 7 bitów każdego bajtu jest używane do zapisu wartości liczby, przy czym pierwsza grupa jest najbardziej znacząca.

Przykłady

Zapis liczby 40: `0010 1000`

Zapis liczby 1622: `1000 1100 0101 0110`

Zapis liczby 884526: `1011 0101 1111 1110 0010 1110`

Kolorem zielonym zaznaczono najbardziej znaczący bit każdego bajtu.

Używane typy danych

Wszelkie liczby kodowane są w systemie U2 oraz zapisane w formacie *little endian*, chyba że zaznaczono inaczej.

Nazwa	Rozmiar (w bajtach)	Opis
BYTE	1	Liczba całkowita z zakresu [-128, 127].
UBYTE	1	Liczba całkowita z zakresu [0, 255].
SHORT	2	Liczba całkowita z zakresu [-32768, 32767].
USHORT	2	Liczba całkowita z zakresu [0, 65535].
INT	4	Liczba całkowita z zakresu [-2147483648, 2147483647].
UINT	4	Liczba całkowita z zakresu [0, 4294967295].
LONG	8	Liczba całkowita z zakresu [-2^{63}, 2^{63} - 1].
ULONG	8	Liczba całkowita z zakresu [0, 2^{64} - 1].
type[N]	N * rozmiar(*type*)	Tablica o stałym rozmiarze zawierająca N elementów typu *type*.
[*type1*, *type2*, …]	rozmiar(*type1*) + rozmiar(*type2*) + …	Struktura z polami typu *type1*, *type2*, itd. W zapisie binarnym nie występują żadne wyrównania, pole typu *type1* występuje jako pierwsze, następnie pole typu *type2*, itd.
VULONG	1-9	Liczba całkowita z zakresu [0, 2^{63} - 1] (dokładny opis w sekcji Typ VULONG).
STRING	2+	Łańcuch znaków. Typ ten składa się z liczby typu VULONG, po której następuje ciąg elementów typu BYTE zakończony elementem o wartości `0x00`. Liczbę tych elementów (włącznie z ostatnim) określa wcześniej wspomniana liczba typu VULONG.

ARRAY<*type*>	1+	Tablica o zmiennym rozmiarze zawierająca elementy typu *type*. Typ ten składa się z liczby typu VULONG, po której następuje ciąg elementów typu *type*. Liczbę tych elementów określa wcześniej wspomniana liczba typu VULONG.
TIMESTAMP	8	Znacznik czasu w formacie Unix Time.

Funkcja MBFSSum

W systemie plików MBFS wykorzystywana jest prosta funkcja skrótu BSD checksum, która dalej nazywana jest MBFSSum.

Zarys systemu plików MBFS

- Sektor to obszar o rozmiarze 512 bajtów, wyrównany do 512 bajtów.
- Klaster to obszar o rozmiarze 2^n bajtów, wyrównany do 2^n bajtów, gdzie $n \geq 9$.
- Cały obszar pamięci sformatowany w systemie MBFS dzielony jest na klastry numerowane kolejnymi nieujemnymi liczbami całkowitymi zaczynając od 0.
- Każdy używany klaster wchodzi w skład jakiegoś pliku.
- Pliki systemowe to pliki przechowujące klastry używane przez system plików.
- Nazwy plików systemowych zaczynają się od znaku *$*. Nazwy te są zarezerwowane przez system plików. Wyjątkiem są nazwy zaczynające się sekwencją *$$* - mogą one być używana do nazywania plików innych niż systemowe.
- Wolne klastry są indeksowane w pliku systemowym *$free*.

Zawartość klastrów

Numer klastra	Nazwa pliku systemowego	Opis
0	$boot	Klaster zawierający sektor rozruchowy. Nieużywany przez system plików.
1	$info	Klaster zawierający strukturę MBFSInfo.
2	$infomirror	Klaster będący kopią klastra 1.
…		Pliki systemowe oraz dane.

Struktura MBFSInfo

Nazwa pola	Offset	Typ	Opis
magic	0	BYTE[4]	Pole zawierające bajty `0x4D 0x42 0x46 0x53` (napis ASCII „MBFS”).
version	4	INT	Wersja systemu plików.
uuid	8	BYTE[16]	Identyfikator.
ccount	24	ULONG	Liczba wszystkich dostępnych klastrów.
root	32	ULONG	Numer pierwszego klastra przechowującego główny plik *$index*.
csp	40	UBYTE	Liczba z przedziału [9, 255] używana do obliczenia rozmiaru klastra ze wzoru 2^{csp}.
iesp	41	UBYTE	Liczba z przedziału [5, csp - 1] używana do obliczenia rozmiaru struktury MBFSIndexElement (opis w dalej części specyfikacji) ze wzoru 2^{iesp}.
sum	42	SHORT	Wynik funkcji MBFSSum, której wejściem są poprzednie pola struktury złączone w jeden ciąg.

Plik $index

Pliki systemowe *$index* zawierają wpisy m.in. z informacjami o plikach (również innych plikach *$index*). Pierwszy wpis każdego pliku *$index* to wpis typu MBFSFileEntry lub MBFSPackedFileEntry, odnoszący się do tego pliku *$index*. Plik *$index* wskazywany w strukturze MBFSInfo przez pole *root* to tzw. główny plik *$index*. Musi on zawierać wpisy typu MBFSFileEntry lub MBFSPackedFileEntry dotyczące plików *$boot*, *$info*, *$infomirror* oraz *$free*.

Każdy wpis podzielony jest na części o stałym rozmiarze, które zostają opakowane w strukturę MBFSIndexElement o następujących polach:

Nazwa pola	Offset	Typ	Opis
next	0	ULONG	Interpretacja zależna od pola *type*.
sum	8	SHORT	Interpretacja zależna od pola *type*.
type	10	BYTE	Typ przechowywanego wpisu.
value	11	BYTE[2^{iesp} - 11]	Dane wpisu. Pole to obejmuje obszar do końca struktury.

Plik *$index* to tablica, której każdy element to struktura MBFSIndexElement. Jeżeli dany wpis mieści się w jednej takiej strukturze, to pole *next* zawiera indeks tego elementu, czyli indeks samego siebie (numeracja od 0). Jeżeli dany wpis nie mieści się jednak w jednej strukturze MBFSIndexElement, to kolejne jego części przechowywane są jako inne elementy tablicy, a ich typ to CONTINUE. Pola *next* zawierają w tym przypadku indeks kolejnego elementu. Ostatni element zawierający wpis musi mieć typ END, a jego pole *next* zawiera indeks pierwszego elementu z danym wpisem. Wszystkie elementy przechowujące jeden wpis muszą znajdować się w tym samym pliku *$index*.

Pole *type* w strukturze MBFSIndexElement może przyjmować następujące wartości:

Wartość	Nazwa	Opis
`0x00`	EMPTY	Nieużywany element. Pozostałe pola struktury MBFSIndexElement są ignorowane.
`0x01`	CONTINUE	Następny element zawierający wpis. Pole *sum* jest ignorowane.
`0x02`	END	Ostatni element z wpisem. Pole *sum* jest ignorowane.
`0x80`	DIR	Tak samo jak dla typu FILE, z tą różnicą, że pole *clusters* w strukturze MBFSFileEntry musi wskazywać klastry zawierające plik *$index*. Wpis taki traktowany jest jako folder, a pole *name* w strukturze MBFSFileEntry traktowane jest jako jego nazwa.
`0x81`	PDIR	Tak samo jak dla typu PFILE, z tą różnicą, że pole *clusters* w strukturze MBFSPackedFileEntry musi wskazywać klastry zawierające plik *$index*. Wpis taki traktowany jest jako folder, a pole *name* w strukturze MBFSPackedFileEntry traktowane jest jako jego nazwa.
`0x82`	FILE	Element zawierający wpis typu MBFSFileEntry. Pole *sum* zawiera wynik funkcji MBFSSum, której wejściem jest pole *name* struktury MBFSFileEntry.
`0x83`	PFILE	Element zawierający wpis typu MBFSPackedFileEntry. Pole *sum* zawiera wynik funkcji MBFSSum, której wejściem jest pole *name* struktury MBFSPackedFileEntry.

System operacyjny może umieścić własne informacje w polu *sum* w przypadku, gdy jest ono ignorowane w danym wpisie.

Wpisy typu MBFSFileEntry i MBFSPackedFileEntry przechowują informacje o pojedynczym pliku i mają następującą strukturę:

Nazwa pola	Typ	Opis
size	VULONG	Rozmiar pliku.
name	STRING	Nazwa pliku.
clusters	**dla MBFSFileEntry:** ARRAY<ULONG> **dla MBFSPackedFileEntry:** ARRAY<[ULONG, ULONG]>	**dla MBFSFileEntry:** Uporządkowana lista numerów klastrów, które zawierają dane pliku. **dla MBFSPackedFileEntry:** Uporządkowana lista par przechowujących informacje o grupach klastrów z danymi pliku. Klastry należące do grupy muszą następować po sobie. Pierwsza liczba pary to numer pierwszego klastra grupy, a druga to liczba klastrów w grupie.
extensions	ARRAY<[STRING, ARRAY<BYTE>]>	Pole z dodatkowymi właściwościami pliku. Pierwszy element pary to nazwa właściwości, a drugi to jej wartość. System operacyjny może przechowywać w tym miejscu dodatkowe informacje takie jak prawa dostępu, właściciel pliku, itp.

Plik $free

Plik *$free* to plik systemowy przechowujący informacje o wolnych klastrach. Jego format jest następujący:

Nazwa	Offset	Typ	Opis
last	0	ULONG	Numer ostatniego używanego klastra.
size	8	ULONG	Rozmiar tablicy przechowującej numery wolnych klastrów.
free	16	ULONG[]	Posortowana rosnąco tablica przechowująca numery wolnych klastrów przed ostatnim używanym klastrem. Obejmuje ona pozostałą przestrzeń pliku.

Klastry z danymi

Dane pliku przechowywane są w klastrach, które nie mogą zostać podzielone na mniejsze części. Klastry wybierane do zapisu danych powinny być możliwie jak najbliżej początku systemu plików, tj. niedopuszczalny jest zapis danych do klastra numer 1000 jeżeli ostatni zajęty klaster to klaster numer 100. Do zapisu danych jednego pliku powinny być, w miarę możliwości, wybierane klastry zaraz obok siebie. Jeżeli dane pliku zapisane są w takich grupach, umożliwi to użycie struktury MBFSPackedFileEntry do zapisu numerów klastrów, a to może przełożyć się na zmniejszenie zapełnienia dysku.

Możliwości oraz rozszerzenia systemu

System MBFS pozwala na przyspieszenie wyszukiwania plików oraz przyspieszenie sprawdzania, czy plik o danej nazwie już istnieje w danym folderze. W tym celu należy zadziałać funkcją MBFSSum na daną nazwę, a następnie porównać wynik z polami *sum* tych struktur MBFSIndexElement, których pole *type* to DIR, PDIR, FILE lub PFILE. W przypadku niezgodności pewnym jest, że plik ma inną nazwę. Jeśli jednak wartości są takie same, to należy porównać całe nazwy.

Możliwe jest także zdefiniowanie dodatkowych typów elementów MBFSIndexElement. Typy takie mogą przykładowo informować, że dany wpis zawiera plik skompresowany, zaszyfrowany lub jest to link symboliczny. Wszelkie tego typu zmiany muszą zostać zasygnalizowane poprzez zmianę wartości pola *magic* w strukturze MBFSInfo na `0x4D 0x42 0x46 0x58` (napis ASCII „MBFX”).