

## Zadanie 5

**Otwarto:** czwartek, 25 maja 2023, 00:00

**Wymagane do:** piątek, 16 czerwca 2023, 23:59

# Blokowanie plików przez użytkowników na wyłączność

Celem zadania jest rozszerzenie serwera *vfs* o mechanizm umożliwiający użytkownikom blokowanie na wyłączność dostępu do wybranych plików. W odróżnieniu od standardowych blokad *flock*, mechanizm ten będzie obligatoryjny (ang. *mandatory*) i będzie działał na poziomie użytkowników, nie procesów. W odróżnieniu od standardowych uprawnień dostępu do plików, mechanizm ten będzie implementował tymczasowe blokowanie, niewymagające zmian atrybutów plików.

## VFS

VFS (ang. *Virtual File System*, pol. *Wirtualny System Plików*) to podsystem systemu operacyjnego umożliwiający jednolity dostęp do plików umieszczonych na różnych systemach plików. Jest on warstwą pośredniczącą między aplikacjami a podsystemami implementującymi konkretne systemy plików (MFS, ext2, procfs itd.). Przetwarza wywołania systemowe realizujące operacje na plikach, implementuje akcje wspólne dla różnych systemów plików oraz przekazuje żądania do odpowiednich systemów plików. Zarządza on także wszystkimi używanymi w systemie plikami i wszystkimi zamontowanymi systemami plików.

?

W [MINIX](#)-ie wirtualny system plików jest zaimplementowany jako serwer *vfs*. Więcej o jego budowie i sposobie działania można przeczytać na Wiki [MINIX](#)-a: [VFS internals](#).

## Wywołania systemowe **VFS\_FEXCLUSIVE** i **VFS\_EXCLUSIVE**

Mechanizm blokowania plików opiera się na nowych wywołaniach systemowych **VFS\_FEXCLUSIVE** i **VFS\_EXCLUSIVE** obsługiwanych przez serwer *vfs*. Za ich pomocą użytkownik może tymczasowo zablokować innym użytkownikom możliwość wykonania na wskazanym pliku następujących działań: otwarcia pliku (wywołania systemowe **VFS\_OPEN** i **VFS\_CREAT**), odczytu (**VFS\_READ**), zapisu (**VFS\_WRITE**), skrócenia (**VFS\_TRUNCATE** i **VFS\_FTRUNCATE**), przeniesienia i zmiany nazwy (**VFS\_RENAME**, zarówno gdy zablokowany plik jest pierwszym, jak i drugim argumentem) oraz usunięcia pliku (**VFS\_UNLINK**). Użytkownik, który zablokował plik może wykonywać te operacje bez ograniczeń z różnych procesów. Natomiast próba wykonania ich przez innego użytkownika powinna zakończyć się błędem **EACCES**.

Argumentami wywołania systemowego **VFS\_FEXCLUSIVE** są deskryptor i flaga oznaczająca wykonywaną akcję. Obsługiwane są następujące akcje:

- **EXCL\_LOCK**: Blokuje plik wskazany przez deskryptor na wyłączność użytkownika, który wywołuje to wywołanie systemowe. Jeśli plik nie zostanie odblokowany jawnie przez użytkownika, to plik zostanie odblokowany automatycznie w momencie, gdy zamknięty zostanie deskryptor, który był argumentem wywołania blokującego plik.
- **EXCL\_LOCK\_NO\_OTHERS**: Działa tak jak **EXCL\_LOCK**, ale plik jest blokowany wyłącznie, gdy nie jest w tym momencie otwarty także przez innego użytkownika (użytkownik blokujący plik może mieć ten plik otwarty dowolną liczbę razy). W przeciwnym przypadku wywołanie kończy się błędem **EAGAIN**.
- **EXCL\_UNLOCK**: Odblokowuje plik wskazany przez deskryptor. Plik odblokować może tylko użytkownik, który go zablokował.
- **EXCL\_UNLOCK\_FORCE**: Odblokowuje plik wskazany przez deskryptor. Plik odblokować może użytkownik, który go zablokował, użytkownik, który jest właścicielem tego pliku lub superużytkownik (ang. *root*, użytkownik o **UID** = 0).

Argumentami wywołania systemowego **VFS\_EXCLUSIVE** są ścieżka do pliku i flaga oznaczająca wykonywaną akcję. Obsługiwane są następujące akcje:

- **EXCL\_LOCK**: Blokuje plik wskazany przez podaną ścieżkę na wyłączność użytkownika, który wywołuje to wywołanie systemowe. Plik pozostaje zablokowany, aż nie zostanie jawnie odblokowany przez użytkownika. Wyjątkiem od tej reguły jest sytuacja, gdy zablokowany plik zostanie przez użytkownika usunięty (wywołaniem systemowym

`VFS_UNLINK`) lub zastąpiony innym plikiem (zablokowany plik będzie drugim argumentem `VFS_RENAME`). W takim przypadku plik zostanie automatycznie odblokowany w momencie, gdy plik przestanie być używany przez wszystkich użytkowników (żaden proces nie będzie miał otwartego tego pliku).

- `EXCL_LOCK_NO_OTHERS`: Działa tak jak `EXCL_LOCK`, ale plik jest blokowany wyłącznie, gdy nie jest w tym momencie otwarty także przez innego użytkownika (użytkownik blokujący plik może mieć ten plik otwarty dowolną liczbę razy). W przeciwnym przypadku wywołanie kończy się błędem `EAGAIN`.
- `EXCL_UNLOCK`: Odblokowuje plik wskazany przez podaną ścieżkę. Plik odblokować może tylko użytkownik, który go zablokował.
- `EXCL_UNLOCK_FORCE`: Odblokowuje plik wskazany przez podaną ścieżkę. Plik odblokować może użytkownik, który go zablokował, użytkownik, który jest właścicielem tego pliku lub superużytkownik (ang. *root*, użytkownik o `UID = 0`).

Mechanizm blokowania plików działa według następującej specyfikacji:

1. Blokowane są wyłącznie zwykłe pliki. Próba zablokowania katalogu, pseudourządzenia, potoku (ang. *pipe*, *fifo*) itp. kończy się błędem `EFTYPE`.
2. Blokowany lub odblokowywany jest konkretny plik, wskazany przez deskryptor lub ścieżkę w momencie blokowania lub odblokowywania go. Plik nie przestaje być zablokowany wskutek późniejszego przeniesienia go (w obrębie partycji), zmiany mu nazwy, wskazania go przez ścieżkę z dowiązaniem (ang. *link*) itd. Technicznie blokowany jest konkretny v-węzeł lub i-węzeł (ang. *v-node/i-node*).
3. Blokady plików nie są zachowywane po odmontowaniu systemu plików. Obecność zablokowanych plików nie uniemożliwia odmontowania systemu plików (jeśli odmontowanie partycji bez zablokowanych plików by się powiodło, to powiedzie się także odmontowanie tej partycji, gdy są na niej zablokowane pliki).
4. Blokada obowiązuje od momentu zablokowania pliku do momentu jego odblokowania. Wymienione wyżej wywołania systemowe sprawdzają możliwość dostępu do pliku w każdym ich wywołaniu. Możliwa jest zatem sytuacja, gdy użytkownik z powodzeniem otworzy plik, a następnie inny użytkownik go zablokuje, więc następna operacja pierwszego użytkownika (np. odczyt zawartości pliku) zakończy się błędem `EACCES`, a gdy drugi użytkownik odblokuje plik, kolejna operacja pierwszego użytkownika (np. ponowna próba odczytu zawartości pliku) zakończy się sukcesem.
5. Użytkownik jest identyfikowany przez jego rzeczywisty identyfikator (ang. *real UID*), niezależnie od wartości identyfikatora efektywnego (ang. *effective UID*).
6. Za pomocą `VFS_FEXCLUSIVE` zablokować plik można wyłącznie, gdy podany deskryptor jest otwarty w trybie do odczytu lub zapisu. W przeciwnym przypadku wywołanie systemowe kończy się błędem `EBADF`. Za pomocą `VFS_EXCLUSIVE` zablokować plik można wyłącznie, gdy użytkownik wywołujący ma uprawnienia do odczytu lub

zapisu pliku wskazanego przez podaną ścieżkę. W przeciwnym przypadku wywołanie systemowe kończy się błędem `EACCES`.

7. Jeśli wywołania systemowe `VFS_FEXCLUSIVE` i `VFS_EXCLUSIVE` nie mogą zakończyć się powodzeniem, kończą się odpowiednim błędem. Na przykład: `EINVAL` – jeśli w wywołaniu podana została inna flaga niż obsługiwane akcje lub wskazany do odblokowania plik nie jest zablokowany, `EBADF` – jeśli podany w wywołaniu deskryptor jest nieprawidłowy, `EALREADY` – jeśli wskazany do zablokowania plik jest już zablokowany, `EPERM` – jeśli użytkownik nie jest uprawniony do odblokowania wskazanego pliku itd.
8. Jednocześnie zablokowanych jest co najwyżej `NR_EXCLUSIVE` plików (stała ta jest zdefiniowana w załączniku do zadania). Wywołania, których obsługa oznaczałaby przekroczenie tego limitu, powinny zakończyć się błędem `ENOLCK`.
9. Wywołania systemowe `VFS_FEXCLUSIVE` i `VFS_EXCLUSIVE` realizują ten sam mechanizm blokowania plików. Możliwe jest zablokowanie pliku, używając jednego wywołania systemowego i odblokowanie tego pliku go za pomocą drugiego wywołania systemowego.

## Załączniki do zadania

Do zadania załączona jest łątka `zadanie5-szablon.patch`, która definiuje stałe opisane w zadaniu oraz dodaje wywołania systemowe `VFS_FEXCLUSIVE` i `VFS_EXCLUSIVE` realizowane przez funkcje `int do_fexclusive(void)` i `int do_exclusive(void)` dodane do serwera `vfs`.

Do zadania załączone są także przykładowe programy `test-fexclusive.c`, `test-exclusive-lock.c` i `test-exclusive-unlock.c`, które ilustrują użycie wywołań systemowych `VFS_FEXCLUSIVE` i `VFS_EXCLUSIVE` do blokowania innym użytkownikom dostępu do pliku podanego jako argument programu. Przykładowy scenariusz użycia programu `test-fexclusive.c`:

```
# make test-fexclusive
clang -O2 -o test-fexclusive test-fexclusive.c
# touch /tmp/test.txt
# ./test-fexclusive /tmp/test.txt
Blokuję plik...
Wynik VFS_FEXCLUSIVE: 0, errno: 0
Czekam... Naciśnij coś
```

W tym momencie próba otwarcia pliku `/tmp/test.txt` przez innego użytkownika nie powiedzie się:

```
# cat /tmp/test.txt
cat: /tmp/test.txt: Permission denied
```

Kontynuowanie działania programu `test-fexclusive` zamknie plik `/tmp/test.txt`, co spowoduje odblokowanie pliku `/tmp/test.txt` i inni użytkownicy będą mogli z nim pracować.

Przykładowy scenariusz użycia programu `test-exclusive-lock.c`:

```
# make test-exclusive-lock
clang -O2 -o test-exclusive-lock test-exclusive-lock.c
# touch /tmp/test.txt
# ./test-exclusive-lock /tmp/test.txt
Blokuję plik...
Wynik VFS_EXCLUSIVE: 0, errno: 0
#
```

W tym momencie próba otwarcia pliku `/tmp/test.txt` przez innego użytkownika nie powiedzie się:

```
# cat /tmp/test.txt
cat: /tmp/test.txt: Permission denied
```

Plik można odblokować za pomocą programu `test-exclusive-unlock.c`:

```
# make test-exclusive-unlock
clang -O2 -o test-exclusive-unlock test-exclusive-unlock.c
# ./test-exclusive-unlock /tmp/test.txt
Odblokowuję plik...
Wynik VFS_EXCLUSIVE: 0, errno: 0
#
```

Teraz inni użytkownicy mogą znów z nim pracować.

Komentarze wewnątrz kodu przykładowych programów ilustrują także użycie pozostałych flag wywołań systemowych.

## Wymagania i niewymagania

1. Wszystkie pozostałe operacje realizowane przez serwer `vfs`, inne niż opisane powyżej, powinny działać bez zmian.

2. Modyfikacje serwera *vfs* nie mogą powodować błędów w systemie (m.in. *kernel panic*) i błędów w systemie plików (m.in. niespójności danych).
3. Modyfikacje nie mogą powodować zmian w zawartości plików, w atrybutach plików, w strukturze plików, ani w formacie, w jakim dane przechowywane są na dysku.
4. Modyfikacje mogą dotyczyć tylko serwera *vfs* (czyli mogą dotyczyć tylko plików w katalogu `/usr/src/minix/servers/vfs/`), za wyjątkiem modyfikacji dodawanych przez łątkę `zadanie5-szablon.patch`. Nie można zmieniać definicji dodawanych przez tę łątkę.
5. Podczas działania zmodyfikowany serwer nie może wypisywać żadnych dodatkowych informacji na konsolę ani do rejestru zdarzeń (ang. *log*).
6. Można założyć, że rozwiązania nie będą testowane z użyciem dowiązań twardych (ang. *hard links*). Nie będzie też testowane blokowanie plików (programów), które będą potem wykonywane.
7. Rozwiązanie nie musi być optymalne pod względem prędkości działania. Akceptowane będą rozwiązania, które działają bez zauważalnej dla użytkownika zwłoki.

## Wskazówki

1. Implementacja serwera *vfs* nie jest omawiana na zajęciach, ponieważ jednym z celów tego zadania jest samodzielne przeanalizowanie odpowiednich fragmentów kodu źródłowego [MINIX](#)-a. Rozwiązując to zadanie, należy więcej kodu przeczytać, niż samodzielnie napisać lub zmodyfikować.
2. Przykładów, jak w serwerze *vfs* zaimplementować daną funkcjonalność, warto szukać w kodzie implementującym już istniejące wywołania systemowe.
3. W internecie można znaleźć wiele opracowań omawiających działanie serwera *vfs*, np. [slajdy do wykładu Prashant Shenoy'a](#). Korzystając z takich materiałów, należy jednak zwrócić uwagę, czy dotyczą one tej samej wersji [MINIX](#)-a i serwera *vfs*, którą modyfikujemy w tym zadaniu.
4. Do [MINIX](#)-a uruchomionego pod *QEMU* można dołączać dodatkowe dyski twarde. Aby z tego skorzystać, należy:
  - A. Na komputerze-gospodarzu stworzyć plik będący nowym dyskiem, np.: `qemu-img create -f raw extra.img 1M`.
  - B. Podłączyć ten dysk do maszyny wirtualnej, dodając do parametrów, z jakimi uruchamiane jest *QEMU*, parametry `-drive file=extra.img,format=raw,index=1,media=disk`, gdzie parametr `index` określa numer kolejny dysku (0 to główny dysk – obraz naszej maszyny).

- C. Za pierwszym razem stworzyć na nowym dysku system plików mfs: `/dev/c0d<numer kolejny dodanego dysku>`, np. `/sbin/mkfs.mfs /dev/c0d1`.
- D. Stworzyć pusty katalog (np. `mkdir /root/nowy`) i zamontować do niego podłączony dysk: `mount /dev/c0d1 /root/nowy`.
- E. Wszystkie operacje wewnątrz tego katalogu będą realizowane na zamontowanym w tym położeniu dysku.
- F. Aby odmontować dysk, należy użyć polecenia `umount /root/nowy`.

## Rozwiązanie

Poniżej przyjmujemy, że `ab123456` oznacza identyfikator studentki lub studenta rozwiązującego zadanie. Należy przygotować łatkę (ang. *patch*) ze zmianami w sposób opisany w treści zadania 3. Rozwiązanie w postaci łatki `ab123456.patch` należy umieścić w Moodle. Łatka będzie aplikowana do nowej kopii [MINIX](#)-a. **Uwaga:** łatka z rozwiązaniem powinna zawierać także zmiany dodane przez łatkę `zadanie5-szablon.patch`. Należy zadbać, aby łatka zawierała tylko niezbędne różnice.

W celu skompilowania i uruchomienia systemu ze zmodyfikowanym serwerem `vfs` wykonane będą następujące polecenia:

```
cd /
patch -t -p1 < ab123456.patch
cd /usr/src; make includes
cd /usr/src/minix/servers/vfs/; make clean && make && make install
cd /usr/src/releasetools; make do-hdboot
reboot
```

## Ocenianie

Oceniana będą zarówno poprawność, jak i styl rozwiązania. Podstawą do oceny rozwiązania będą testy automatyczne sprawdzające poprawność implementacji oraz przejrzanie kodu przez sprawdzającego. Rozwiązanie, w którym łatka nie nakłada się poprawnie, które nie kompiluje się lub powoduje *kernel panic* podczas uruchamiania, otrzyma 0 punktów.





Wymaganą częścią zadania jest implementacja akcji `EXCL_LOCK` i `EXCL_UNLOCK` wywołania systemowego `VFS_FEXCLUSIVE`. Za poprawną i w dobrym stylu pełną implementację tej części funkcjonalności rozwiązanie otrzyma 3 pkt. Za poprawną i w dobrym stylu pełną implementację akcji `EXCL_LOCK` i `EXCL_UNLOCK` wywołania systemowego

`VFS_EXCLUSIVE` rozwiązanie otrzyma 1,5 pkt. Za poprawną i w dobrym stylu pełną implementację akcji `EXCL_LOCK_NO_OTHERS` i `EXCL_UNLOCK_FORCE` w obu wywołaniach systemowych rozwiązanie otrzyma 0,5 pkt. Próba wykonania wywołania systemowego z flagą oznaczającą akcję, której rozwiązanie nie obsługuje, powinna zakończyć się błędem `ENOSYS`.

## Pytania

Pytania do tego zadania należy kierować na adres [ms383396@students.mimuw.edu.pl](mailto:ms383396@students.mimuw.edu.pl) z [SOzad5] w temacie wiadomości. Odpowiedzi na często zadawane pytania będą pojawiać się na forum w Moodle.

## Załączniki

 <a href="#">test-exclusive-lock.c</a>	24 maja 2023, 17:58
 <a href="#">test-exclusive-unlock.c</a>	24 maja 2023, 17:58
 <a href="#">test-fexclusive.c</a>	24 maja 2023, 17:58
 <a href="#">zadanie5-szablon.patch</a>	24 maja 2023, 17:58

Dodaj pracę

## Status przesłanego zadania

Status przesłanego zadania	Nie przesłano jeszcze zadania
Stan oceniania	Nieocenione
Pozostały czas	Pozostało 4 dni 14 godzin
Ostatnio modyfikowane	-



**Komentarz do przesłanego zadania**[▶ Komentarze \(0\).](#)

Skontaktuj się z nami



Follow us



Skontaktuj się z pomocą techniczną

Jesteś zalogowany(a) jako Stanisław Bitner (Wyloguj)

Podsumowanie zasad przechowywania danych

Pobierz aplikację mobilną

Pobierz aplikację mobilną

Motyw został opracowany przez

conecti.me

