Tłumaczenie z angielskiego: Fibogacci (http://mxlinux.pl, https://facebook.com/mxlinuxpl)
Wersja: 20220627T0125

Oryginał na: https://www.facebook.com/groups/665300826967101/posts/2299873573509810/

Uwagi dotyczące systemu antiX live, używanego przez MX Linux

Te notatki dotyczące systemu antix Live, używanego przez MX Linux, zostały stworzone przez członka oficjalnego Forum MX Linux ***thomasl*** *i są tutaj umieszczone za jego uprzejmą zgodą:*

(Zastrzeżenie: te uwagi są oparte na tym, jak ja rozumiem funkcję ***Persistence*** w MX Linux – to rozumienie może, ale nie musi być prawidłowe. Dlatego jestem wdzięczny za poprawki i/lub dodatkowe materiały od wszystkich, którzy znają tę funkcję. Istnieje również seria filmów autorstwa @dolphin_oracle, które wyjaśniają i pokazują wiele szczegółów, o których piszę w tym i kolejnym wpisie.)

Dla mnie jedną z najlepszych (i niedocenianych w mojej opinii) funkcji MX Linux jest opcja oszczędnej instalacji ***Frugal Install*** (i powiązane z nią opcje ***Persistence*** – utrwalania). Dlaczego? Z wielu powodów - niektóre z nich przedstawię w tym i kolejnym wpisie.

Po pierwsze, oszczędną instalację MX można zredukować do **zaledwie czterech plików**. Tak, zgadza się. Kompletna instalacja Linuksa w zaledwie czterech plikach. Zapisz te pliki na dysku kopii zapasowej (lub dwóch) lub (bezpiecznie zaszyfrowane) w chmurze i masz pełną kopię zapasową systemu. Nawet w pełni funkcjonalna instalacja ze wszystkimi „bajerami” (jak opisano poniżej) jest tylko o jeden plik większa (istnieją również małe, dodatkowe pliki specyficzne dla danego komputera, ale nie są one wymagane do utworzenia kopii zapasowej). Całkiem nieźle, co?

Po drugie, te cztery (lub pięć) plików może znajdować się w **DOWOLNYM** systemie plików. Oznacza to, że możesz mieć pełny system Linux uruchomiony na komputerze z systemem Windows bez konieczności ponownego partycjonowania dysku(ów) lub zmiany systemu plików na żadnej z istniejących partycji. Jest to szczególnie cenne, jeśli chcesz „badać nowe obszary” z MX Linux.

Albo taki przykład. Mogę skopiować mój system (tzn. te cztery lub pięć plików) na pendrive USB i zainstalować go za pomocą kilku poleceń CLI na prawie każdym komputerze (z wystarczającą ilością wolnego miejsca na te pliki). Tak więc wdrożenie i uruchomienie MX na drugim (lub trzecim i kolejnych) komputerze lub laptopie to łatwizna.

Przyjrzyjmy się bliżej tym czterem plikom. Z jednej strony mamy ***vmlinuz*** i ***initrd.gz***, które są dwoma podstawowymi plikami dla GRUB (lub dla innego podobnego menedżera startowego) do uruchomienia systemu z dysku. ***vmlinuz*** to jądro Linuksa, a ***initrd.gz*** zawiera początkowy *ramdisk* używany do ładowania i inicjalizacji pełnego system.

Z drugiej strony mamy ***linuxfs*** i ***rootfs***. Te dwa pliki są zazwyczaj znacznie większe niż ***vmlinuz*** i ***initrd.gz***, ponieważ przechowują "istotę" systemu (mój plik *linuxfs* ma obecnie rozmiar 1,87 GB podczas gdy mój *rootfs* ma zawsze dokładnie 2 GB). ***linuxfs*** zawiera skompresowany system plików tylko do odczytu (*read-only)* z kompletnym systemem operacyjnym - każdy plik niezbędny

do uruchomienia MX Linux oraz dodatkowo pliki użytkownika. W moim przypadku *linuxfs* jest nieco poniżej 2 GB (z kompresją *lz4*), ale używam mocno dostosowanego systemu, z mnóstwem rzeczy, których nie chcę lub nie potrzebuję usuwać.

Tak więc system plików tylko do odczytu ma wiele zalet (na przykład wirus/trojan nie może się tam dostać), ale oczywiście nie może też zapisywać żadnych zmian. W tym miejscu wchodzi do gry drugi plik, ***rootfs***. To jest jeden z dwóch plików ***persistence*** (utrwalania) obsługiwanych przez MX. Zawiera pełny system plików Linux, choć początkowo jest (prawie) pusty.

Jak działa ta funkcja ***Persistence***? Kod inicjujący w ***initrd.gz*** zasadniczo robi trzy rzeczy (tzn. robi dużo więcej, ale te trzy rzeczy dotyczą funkcji *persistence*): najpierw montuje ***linuxfs***, aby zawartość była dostępna tak, jakby była normalną partycją dysku (ale, jak wspomniano, jest to montowanie tylko do odczytu – *read-only*). Następnie montuje kolejny plik, ***rootfs***, do odczytu i zapisu (*read-write*). Wreszcie używa zamontowanego ***rootfs*** do utworzenia warstwy (nakładki w języku technicznym) nad zamontowanym ***linuxfs***, tak aby uzyskiwanie dostępu do systemu plików najpierw przechodziło przez warstwę ***rootfs***, a dopiero następnie do warstwy ***linuxfs***.

Teraz musimy rozdzielić dwa scenariusze: prosty dostęp do odczytu i dostęp do zapisu. Najpierw odczyt: jeśli aplikacja chce odczytać plik, to nakładka przechwytuje to wywołanie i sprawdza, czy ten plik nie znajduje się już w warstwie ***rootfs***. Jeśli znajduje się, to stamtąd jest odczytywany i zwracany. Jeśli nie znajduje się tam, jest wtedy odczytywany z warstwy ***linuxfs*** i zwracany.

Zapisywanie to jednak inna historia: każdy dostęp do zapisu jest przechwytywany i trafia bezpośrednio do warstwy ***rootfs***, tj. ten plik lub zmiana jest zawsze przechowywana w systemie plików ***rootfs***. W zasadzie tak wygląda implementacja *persistence* (utrwalania) w MX Linux. (Istnieją pewne drobne komplikacje, takie jak usuwanie plików, ale podstawowa zasada jest taka, że wszystkie zmiany w uruchomionej, oszczędnej instalacji ***Frugal Install*** z włączonym ***root persistence*** są przechwytywane i przechowywane w warstwie ***rootfs***.)

Istnieją dwa sposoby na zamontowanie pliku ***rootfs***. Pierwszy to montowanie bezpośrednie (tzn. jako urządzenie pętlowe – *loop device*). Oznacza to, że każda przechwycona zmiana jest natychmiast zapisywana do pliku ***rootfs*** i pojawia się na dysku twardym. Ten tryb nazywa się ***static persistence*** (utrwalanie statyczne).

Drugi tryb (zwany ***dynamic persistence*** – utrwalanie dynamiczne) działa poprzez skopiowanie pełnej zawartości ***rootfs*** z dysku do pamięci RAM podczas rozruchu, a następnie zamontowanie tej kopii z pamięci RAM. I znowu, wszystkie dostępy do zapisu i inne zmiany są przechwytywane, ale tym razem są one przechowywane tylko w pamięci RAM, a nie w pliku ***rootfs*** na dysku. Oznacza to, że ***dynamic persistence*** jest szybsze niż ***static persistence*** (zwłaszcza jeśli plik ***rootfs*** znajduje się na wolnej pamięci USB), ale także oznacza, że wszystkie zmiany są przechowywane tylko w pamięci RAM i mogą zostać utracone, na przykład w przypadku awarii zasilania.

Oczywiście oznacza to również, że część pamięci RAM nie jest dostępna dla działającego systemu i czy będzie to problemem zależy od całkowitej ilości zainstalowanej pamięci RAM. W moim produkcyjnym komputerze mam 16 GB i nie mam żadnych problemów, ale komputer z zaledwie 4 GB pamięci RAM prawdopodobnie nie będzie działał zbyt dobrze w trybie ***dynamic persistence***.

Co więcej, w tym trybie (***dynamic persistence***) użytkownik musi upewnić się, że zmiany zostaną rzeczywiście zapisane w pewnym momencie do pliku ***rootfs*** na dysku - jeśli te zmiany mają być trwałe.

Można to zrobić na dwa sposoby: bezpośrednio uruchamiając narzędzie CLI o nazwie „***persist-save***" lub za pomocą narzędzia ***mx-remastercc*** (MX Remaster) (przycisk „*Save root persistence*" - *Zapisz root persistence*). Można to też zrobić podczas wyłączania systemu, albo ręcznie albo automatycznie. Ponownie, zobacz narzędzie ***mx-remastercc*** (MX Remaster) (przycisk „*Configure live persistence" - Konfiguruj live persistence*) lub narzędzie CLI „***persist-config***". Tak czy inaczej, wszystkie skumulowane zmiany przechowywane w pamięci RAM są wtedy zapisywane do pliku ***rootfs*** na dysku twardym w jednym podejściu.

Jak już wspomniałem, ***dynamic persistence*** wymaga komputera z „wystarczającą" ilością pamięci RAM do skopiowania zawartości pliku ***rootfs***. Chociaż z drugiej strony, rzeczywisty rozmiar pliku ***rootfs*** sam w sobie nie jest prawdziwą miarą wymaganej pamięci RAM, ponieważ przydzielane jest tylko miejsce używane przez pliki. Na przykład mam plik ***rootfs*** o pojemności 2 GB, ale obecnie zajmuje tylko 1,2 GB pamięci RAM.

Opcje ***Persistence*** nie kończą się na ***rootfs***. Możliwe jest utworzenie i zamontowanie kolejnego pliku *persistence* o nazwie ***homefs***. Ten plik przechowuje tylko zmiany w zawartości katalogu **/home**, ale poza tym działa podobnie do ***static root persistence***. Oznacza to, że plik ***homefs*** jest montowany jako do odczytu i zapisu (*read-write*), a wszystkie zapisy dla dowolnych plików w **/home** (lub jakakolwiek inna zmiana w **/home**) zostaną przechwycone i natychmiast zachowane w pliku ***homefs*** na dysku. Innymi słowy, ***home persistence*** jest zawsze statyczne.

Jak dotąd wszystko jasne? ;) Ale jak ***rootfs*** i ***homefs*** współdziałają ze sobą? Albo inaczej, jak ostatecznie zachodzą w nich zmiany?

Mamy sześć następujących permutacji:

1. Brak ***persistence***: żadne zmiany nie są nigdzie zapisywane

2. Tylko ***static rootfs***: zmiany (w tym te w **/home**) są zapisywane w ***rootfs*** na dysku, w momencie, gdy się pojawiają

3. Tylko ***dynamic rootfs***: zmiany (włącznie z tymi w **/home**) są zapisywane w kopii RAM ***rootfs*** tak jak one się pojawiają i muszą zostać zapisane na dysku albo ręcznie, albo przy wyłączaniu systemu

4. Tylko ***homefs***: zmiany w **/home** są zapisywane w ***homefs*** na dysku na bieżąco; inne zmiany nie są zapisywane

5. ***static rootfs*** i ***homefs***: zmiany (z wyjątkiem tych w **/home**) są zapisywane w ***rootfs*** na bieżąco; zmiany w **/home** są zapisywane w ***homefs*** na dysku, w momencie, gdy się pojawiają

6. ***dynamic rootfs*** i ***homefs***: zmiany (z wyjątkiem tych w **/home**) są zapisywane w kopii RAM ***rootfs*** w momencie, gdy się pojawiają i muszą zostać zapisane albo ręcznie, albo przy zamykaniu systemu; zmiany w **/home** są zapisywane w ***homefs*** na dysku w momencie, gdy się pojawiają.

Przyjrzyjmy się więc, co się dzieje, gdy tworzymy (lub zmieniamy) dwa różne pliki, jeden w **/usr**, a drugi w katalogu domowym użytkownika dla tych sześciu trybów:

```
$ sudo touch /etc/pliktestowy; touch ~/pliktestowy
```

1. Żaden z tych plików nie jest zapisywany.

2. Oba pliki trafiają do pliku ***rootfs*** na dysku.

3. Oba pliki trafiają do kopii ***rootfs*** w pamięci RAM i musi ona zostać zapisana (kopia, na dysku), albo ręcznie albo podczas zamykania systemu. Jeśli tak się stanie, oba pliki trafiają do pliku ***rootfs*** na dysku.

4. **~/pliktestowy** jest zapisywany w pliku ***homefs*** na dysku. **/etc/pliktestowy** nie jest zapisywany.

5. **~/pliktestowy** jest zapisywany w pliku ***homefs*** na dysku. **/etc/pliktestowy** jest zapisywany w pliku ***rootfs*** na dysku.

6. **~/pliktestowy** jest zapisywany w pliku ***homefs*** na dysku. **/etc/pliktestowy** jest zapisywany w kopii ***rootfs*** w pamięci RAM i musi ona zostać zapisana (kopia, na dysku), albo ręcznie albo podczas zamykania systemu. Jeśli tak się stanie, plik ten trafi do pliku ***rootfs*** na dysku.

Przez 99% czasu używam tylko dwóch z tych opcji. Mój „normalny” tryb pracy to tryb 6, więc zmiany w **/home** są zapisywane natychmiast, a wszystkie inne zmiany są przechowywane w kopii ***rootfs*** w pamięci RAM. Jednak NIGDY nie zapisuję żadnej z tych zmian na dysku ręcznie ani przy zamykaniu systemu, więc gdy ponownie uruchamiam system w trybie 6, plik ***rootfs*** na dysku nigdy nie zmienia się między kolejnymi uruchomieniami.

Otwiera to ciekawe możliwości, m.in. dla testowania różnych rzeczy lub testowego instalowania oprogramowania. Oznacza to również, że wirus lub inne dziadostwo nie może naprawdę uszkodzić zawartości pliku ***rootfs*** na dysku. Ta konfiguracja przetrwa nawet "sudo rm rf /usr" lub podobne takie katastrofy w terminalu.

Jeśli i kiedy chcę coś zmienić poza **/home** (np. zrobić aktualizację systemu, zainstalować aplikację itp.), najpierw zapisuję bieżącą wersję pliku ***rootfs*** na nośniku kopii zapasowej, a następnie (ponownie) uruchamiam komputer w trybie 5, dokonuję wszystkich planowanych zmian, które chcę wykonać (a które od razu pojawiają się w pliku ***rootfs*** na dysku) i na koniec restartuję komputer z powrotem w trybie 6. Jeśli wszystko poszło dobrze, po prostu kontynuuję pracę, ale w razie problemu mogę po prostu skopiować kopię zapasową ***rootfs***, którą stworzyłem, zanim zacząłem aktualizować/zmieniać system. Nie robię tego zbyt często, mniej więcej raz lub dwa razy w miesiącu.

(Możesz zadać sobie pytanie, dlaczego nie dokonuję zmian podczas działania ***dynamic persistence*** (tj. tryb 6) i następnie po prostu nie uruchamiam "***persist-save***". To dlatego, że miałem powtarzalne problemy z zapisywaniem zmian w plikach podczas działania ***dynamic persistence***, więc przestałem używać tej funkcji (więcej o tym na: https://forum.mxlinux.org/viewtopic.php?t=56823). Jest całkiem możliwe, że ten problem został rozwiązany w nowszych wersjach MX, ale

nigdy tego nie sprawdzałem, ponieważ moja obecna konfiguracja jest solidna i stabilna. Twoje doświadczenie może być odmienne.

Czasami chcę po prostu zmienić pojedynczy plik (powiedzmy w **/etc**) lub po prostu dodać kilka plików. W tym przypadku niekoniecznie zadaję sobie trud ponownego uruchomienia w trybie 5.

Zamiast tego zwyczajnie montuję plik ***rootfs***:

```
$ sudo mount -o loop /live/boot-dev/<TWOJ-KATALOG-MXLINUX>/rootfs /mnt/rootfs
```

zmieniam plik(i), które chcę zmienić i odmontowuję **/mnt/rootfs**. Na przykład, gdybym chciał zmienić **/etc/environment**, wyedytowałbym **/etc/environment** jako *root*, zamontowałbym plik ***rootfs***, jak pokazałem powyżej, a następnie skopiowałbym **/etc/environment** do **/mnt/rootfs/upper/etc/environment** jako *root* i następnie odmontowałbym (jednakże ma to sens tylko w przypadku ***dynamic rootfs persistence***, a NIE ze ***static rootfs persistence***). Rzadko jednak tak robię.

Ta konfiguracja udowodniła swoją solidność (używana przez ponad 4 lata, zaczynając od MX18) i mała niedogodność związana z koniecznością sporadycznego uruchamiania w trybie ***static persistence***, aby dokonać aktualizacji i instalacji programów, jest dla mnie warta wynikającej z tego dodatkowej stabilności i spokoju umysłu. Ale oczywiście oszczędne instalacje (*Frugall Installs*) MX Linux i ich opcje *persistence* (utrwalania) są tak elastyczne, że jest to tylko jeden z wielu możliwych sposobów na zrobienie tego i inni mogą znaleźć całkowicie odmienny zestaw opcji, znacznie lepiej dostosowany do ich potrzeb.

Co zatem w przypadku, gdy po wielu tygodniach lub miesiącach w pliku ***rootfs*** pojawiło się tyle zmian, że nie zostało już wiele wolnego miejsca? Dla mojego pliku ***rootfs*** o rozmiarze 2 GB, osiągnięcie punktu, w którym wykorzystane jest trochę ponad 1,6 GB zajmuje zwykle około czterech do sześciu miesięcy, a rozwiązanie jest szybkie i łatwe.

Tworzę po prostu pełną migawkę (*snapshot*) systemu za pomocą narzędzia MX Zrzut systemu (*MX Snapshot*) (używając jego opcji „Zachowaj konta”). A migawka jest w rzeczywistości plikiem ISO ze wszystkimi plikami wymaganymi do uruchomienia pełnego systemu i zawiera, w katalogu /antiX, zupełnie nowy plik ***linuxfs***, który zawiera wszystkie zmiany zgromadzone w dodanym pliku ***rootfs***.

Więc w zasadzie otwieram ten plik ISO, kopiuję nowy ***linuxfs*** (i ***initrd.gz***, jeśli się zmienił) do mojego katalogu startowego MX, aby zastąpić stary ***linuxfs*** i usunąć stary plik ***rootfs*** (ponieważ nie jest już więcej potrzebny). Następnie uruchamiam system w trybie 5, MX init (proces uruchomieniowy) rozpoznaje, że obecnie nie istnieje plik ***rootfs*** i pozwala mi stworzyć nowy, pusty.

Jak wspomniałem, mój plik ***rootfs*** ma zawsze rozmiar 2 GB, chociaż rzeczywista pamięć RAM wymagana do uruchomienia ***dynamic persistence*** będzie mniejsza, ponieważ tylko używana przestrzeń w pliku ***rootfs*** potrzebuje miejsca w pamięci RAM. Plik ***homefs*** pozostaje przez cały czas nietknięty i będzie nadal działał z nowym plikiem ***linuxfs*** i ***rootfs***.

Oczywiście tworzenie i używanie migawek pozwala na znacznie więcej niż pokazuje ten krótki opis, ale to jest prawdopodobnie dobry temat na osobną notatkę.

Jedna fajna sztuczka z ***persistence*** związana jest z różnicą między ***rootfs*** i ***homefs***. Jak opisałem powyżej, jeśli pracujesz zarówno z ***root***, jak i ***home persistence*** (tj. tryb 5 lub 6 w tej notatce), wtedy wszystkie zmiany w **/home** trafiają do ***homefs***, podczas gdy wszystkie inne zmiany trafiają do ***rootfs*** (jeśli jest statyczny lub zapisany).

Więc jeśli dodasz plik do **/home/$USER** wtedy ten nowy plik znajdzie się w ***homefs***, ale nie zostanie zapisany w wersji ***rootfs*** **/home/$USER**. I dlaczego miałby być zapisywany... w końcu jest w ***homefs***. Jeśli jednak system zostanie uruchomiony w trybie 2 lub 3 (tj. z wyłączonym ***home persistence***), to wszystkie te pliki i zmiany w **/home**, które zgromadziły się w pliku ***homefs*** nie będą obecne, ponieważ ***home persistence*** nie jest aktywne.

Ale dlaczego ktoś chciałby uruchamiać system w trybie 2 lub 3? Cóż, bardzo rzadko, kiedy chcę coś przetestować naprawdę, NAPRAWDĘ niepewnego i ryzykownego, chcę uruchomić system, który jest całkowicie bezpieczny od WSZELKICH zmian. Tryb 6 nie pasuje do tego scenariusza, ponieważ zapobiegnie zmianom w ***rootfs***, ale nie w **/home**, ponieważ te są zapisywane w pliku ***homefs***, na bieżąco. A całkowicie odizolowany system można uzyskać, uruchamiając go w trybie 3, w którym ***root persistence*** jest dynamiczne, a ***home persistence*** jest wyłączone. Ale… jeśli ***home persistence*** jest wyłączone, wszystkie zmiany zapisane w ***homefs*** nie są widoczne! A nie tego chcę.

Na szczęście istnieje sposób na przeniesienie wszystkich dokonanych zmian w **/home** przechowywanych w pliku ***homefs*** z powrotem do pliku ***rootfs*** - to doskonale pokazuje elastyczność systemu utrwalania (*persistence)* MX. To nie jest trywialne zadanie, ale raz opanowane, działa bardzo dobrze.

Zasadniczo raz w miesiącu uruchamiam system w trybie 5 (tzn. ***static root persistence*** i ***home persistence***). W tym trybie plik ***homefs*** jest montowany w **/home**. Natomiast nakładka dla pliku ***rootfs*** jest montowana w **/live/aufs** i zmiany w pozostałej części systemu natychmiast trafiają do ***rootfs***. Tak więc z pomocą odpowiedniego programu do synchronizacji (np. FreeFileSync lub podobnego) mogę teraz po prostu zaktualizować katalog **/live/aufs/home/$USER** (który znajduje się w ***rootfs***) z zawartością mojego katalogu **/home/$USER**

Powiedzmy, że mam plik **/home/$USER/pliktestowy**, którego jeszcze nie ma w ***rootfs***: jeśli został skopiowany z **/home/$USER/pliktestowy** do **/live/aufs/home/$USER/pliktestowy**, jest teraz dostępny również w ***rootfs***. I tak dalej. Wielką zaletą, po wykonaniu synchronizacji z ***homefs*** do ***rootfs***, jest to, że mogę wtedy uruchomić system w trybie 3 (tzn. ***dynamic root persistence*** i BEZ ***home persistence***) i mam wszystkie pliki, w tym wszystkie w ***homefs***, do mojej dyspozycji. Teraz żadne zmiany, ani w **/home**, ani w innych częściach systemu, nie są zapisywane na dysku (o ile nie zostaną zapisane przez użytkownika). System uruchomiony w ten sposób przetrwa wszystko, nawet "sudo rm -rf /".

Aby to wszystko podsumować, oto ostatnia sztuczka, która może być warta poznania. To całkowicie możliwe, aby mieć więcej niż jeden zestaw plików ***persistence*** ("zestaw" oznacza w tym przypadku jeden lub obydwa rodzaje tych plików) i uruchamiać komputer do dwóch, trzech lub więcej zupełnie różnych systemów. Można to zrobić za pomocą opcji rozruchu „**pdir=**”.

Zasadniczo ta opcja mówi procedurom utrwalania (*persistence*), gdzie szukać pliku ***rootfs*** i/lub plik(ów) ***homefs***.

Możesz więc mieć jeden katalog MX ze swoimi "normalnymi" plikami systemowymi (tzn. ***vmlinuz***, ***initrd.gz***, ***linuxfs*** i pliki ***persistence***), a następnie inny katalog (lub nawet dwa lub więcej) z zupełnie innym zestawem plików ***persistence*** (i tylko te są wymagane, a nie żadne z pozostałych plików). To może się przydać, jeśli chcesz przetestować jakieś oprogramowanie przez dłuższy czas lub jeśli chcesz coś zainstalować, ale nie chcesz mieć tego w swoim „normalnym” systemie.

Na przykład, aby utworzyć drugi system z zainstalowanym GIMP-em, możesz utworzyć katalog, powiedzmy GIMP, poniżej katalogu, w którym znajduje się twoja oszczędna instalacja MX (*Frugall Install*). Następnie skopiujesz bieżący plik ***rootfs*** i/lub plik(i) ***homefs*** do katalogu GIMP (najlepiej nie robić tego podczas gdy system uruchomiony jest w trybie oszczędnej instalacji (*Frugall Install*), ale korzystając z systemu Live USB na pendrivie, ponieważ plik ***homefs*** cały czas się zmienia).

Następnie uruchomisz system w wersji „GIMP”, tworząc nowy wpis rozruchowy, dodając „**pdir=<TWOJ-KATALOG-MXLINUX>/GIMP**” do opcji rozruchu. Następnie instalujesz GIMP (który trafia do pliku ***rootfs*** w <TWOJ-KATALOG-MXLINUX>/GIMP) i kiedy chcesz pracować z wersją systemu GIMP, po prostu uruchamiasz system z tym plikiem ***rootfs***. Twój główny system pozostaje całkowicie nietknięty.