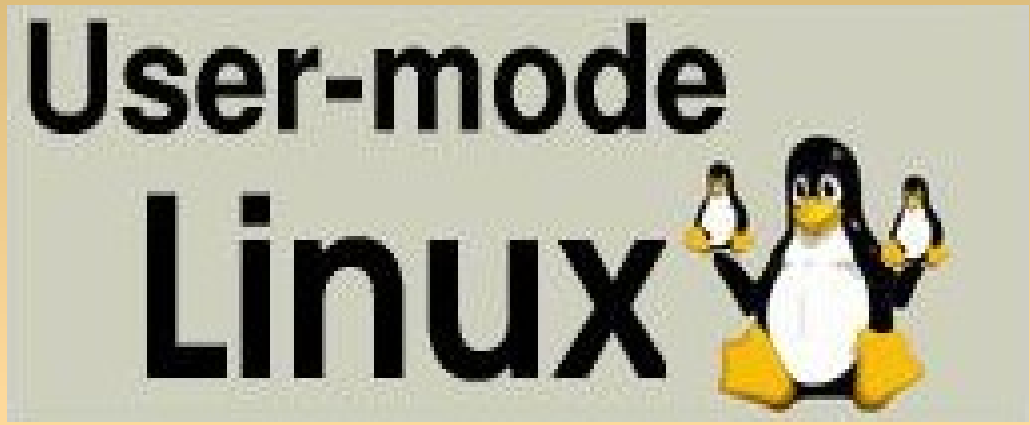
UML

UML



http://user-mode-linux.sourceforge.net/

System operacyjny Linux
zagnieżdżony w zewnętrznym
systemie operacyjnym (Linux)

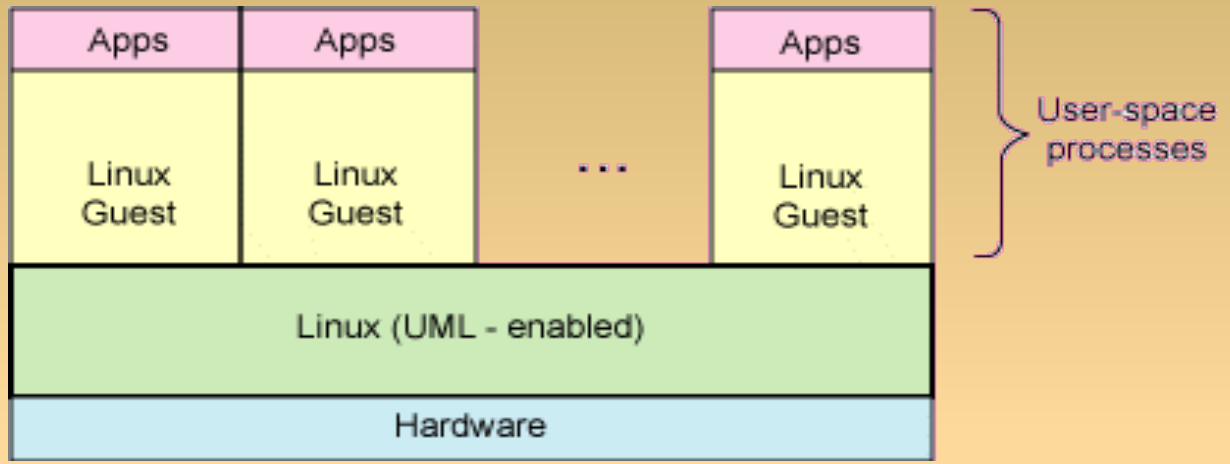
UML

- Autor: Jeff Dike
- Koncepcja powstała w 1999 r.
- Początkowo jako patch do jądra 2.0
- Oryginalnie projektowany jako narzędzie developerów, mające przyspieszyć rozwijanie oprogramowania i zredukować wpływ wymagań sprzętowych
- Pierwsza wersja wydana w 2000 r.

UML

- Obecnie UML jest w jądrze systemu Linux jako jedna z architektur (zintegrowany z jądrem Linuxa od wersji 2.6.0)
- Obsługiwane platformy: oryginalnie utworzony dla architektury x86, przeniesiony na IA-64 i PowerPC
- Licencja: GPL ver. 2

UML



- Nie komunikuje się ze sprzętem, jest od niego oddzielony warstwą macierzystego systemu – hosta
- Korzysta z udostępnianych przez host funkcji systemowych
- Przez hosta traktowany jest jako zwykły proces, przez uruchomione w nim aplikacje – jako jądro
- Jest przykładem wirtualizacji na poziomie systemu oper.

UML

- Ma własne, niezależne zasoby i urządzenia (wirtualne) – dyski twarde, pamięć, kartę sieciową
- Nie tworzy kompletnego wirtualnego hardware'u – raczej działa na zasadzie przechwytywania odwołań systemowych i przekierowywania ich do jądra hosta
- Umożliwia uruchomienie wielu wirtualnych komputerów pracujących pod kontrolą systemu - hosta, które są odizolowane zarówno od siebie, jak i od systemu macierzystego (wraz z kontrolowanym przez niego sprzętem)

UML

Zastosowania:

- testowanie i debugowanie nowych wersji jądra (inne debbugery jądra – np. kgdb – wymagają dwóch maszyn)
- testowanie nowych dystrybucji
- uruchomienie systemu z prawami roota na koncie użytkownika
- bezpieczne testowanie niepewnych / niebezpiecznych programów (sandbox)

Zastosowania c.d.:

- uruchamianie poszczególnych usług sieciowych (w przypadku włamania brak dostępu do hosta - o ile nie jest wkompilowane wsparcie dla hostfs - dostępu do głównego systemu plików hosta)
- zastosowania komercyjne: wirtualne serwery (klient ma złudzenie własnego systemu oraz jest odizolowany od innych klientów) - niższe koszta (na jednym serwerze można uruchomić wiele “serwerów” klientów)

UML

Zastosowania c.d.:

- tworzenie honeypot'ów (pułapek mających na celu wykrycie prób nieautoryzowanego użycia systemu czy pozyskania danych)
- testowanie działania systemu pod różnymi konfiguracjami sprzętowymi
- jednoczesne korzystanie z kilku dystrybucji linuxa
- symulowanie wirtualnych zasobów, np. pamięci
- badania, nauka i zabawa

UML

Tryby działania:

- Tracing Thread (TT)
- Separate Kernel Address Space (SKAS) – poprawa wydajności i bezpieczeństwa

UML

Tryb TT:

- oparty na wątku śledzącym
- jądro znajduje się w górnej przestrzeni adresowej procesów, dostępne do odczytu i zapisu (wspólna przestrzeń adresowa - mniejsze bezpieczeństwo)
- dla każdego procesu w wirtualnym systemie tworzony jest odpowiadający jeden proces w systemie-hoście
- korzysta z funkcji systemowej `ptrace` do przechwytywania sygnałów
- przy wywołaniach systemowych obsługa przekazywana jest za pomocą sygnałów

UML

Tryb SKAS:

- oddzielna (niewidoczna) przestrzeń adresowa jądra
- w hoście tworzone są zawsze tylko 4 procesy na potrzeby wirtualnego systemu
- wyeliminowanie obsługi sygnałów - poprawia wydajność
- wymaga odpowiedniej konfiguracji jądra systemu-hosta
- trudniejszy w debuggowaniu

UML

UML Utilities:

- *mconsole* – interfejs jądra umożliwiający dynamiczne dodawanie sprzętu, zatrzymywanie jądra, tworzenie backupów
- *mkcow* – tworzy obraz dysku (copy-on-write)
- *moo* – łaczy obraz dysku z wersją zapasową
- *net* – ułatwia konfigurację sieci w obrazie dysku

UML

Cechy:

- nie emuluje całej architektury
- dostępny, darmowy kod
- szybkość działania – kompilowany do kodu maszynowego
- ciężar dostosowania sprzętowego na systemie – hoście
- wolniejszy od podobnych narzędzi (XEN, OpenVZ)
- prosty w instalacji (tryb TT) i użytkowaniu
- nie wymaga zgodności wersji

UML

Cechy c.d.:

- możliwe jest uruchomienie UML-a na UML-u
- można podpiąć UML-a pod strukturę sieci, przez co będzie on z zewnątrz widziany jako normalny komputer
- można emulować nieistniejące zasoby (RAM, procesory, urządzenia IO)

UML

- Konkurencyjne projekty (m.in.): Linux on Linux, Windows on Linux, Linux on Windows