

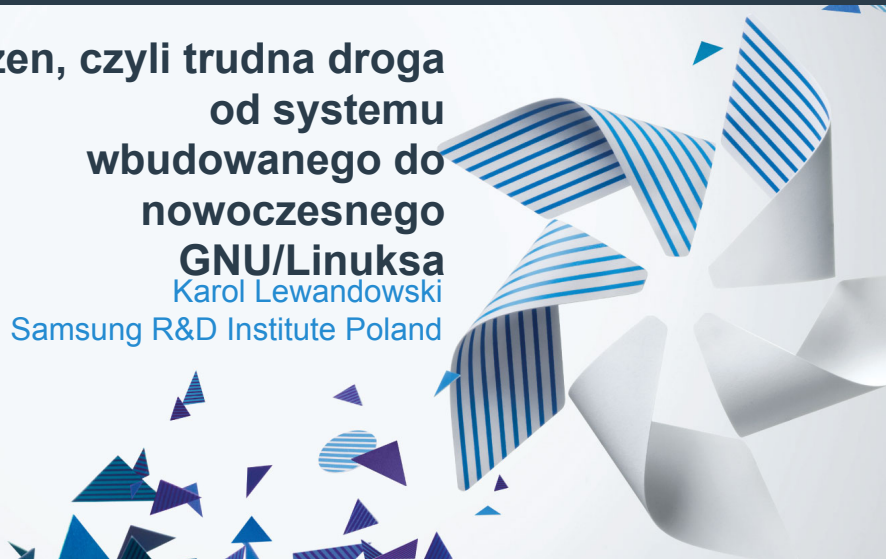
jesień linuxowa | jesien.org

11-13 Październik 2013 - Szczyrk, Polska

Tizen, czyli trudna droga od systemu wbudowanego do nowoczesnego GNU/Linux

Karol Lewandowski

Samsung R&D Institute Poland



Outline

Wstęp

Historia

Problem 1: Budowa oprogramowania

Problem 2: init(8)

Problem 3: Niezawodność systemu

Dalsze kroki

Q&A



Wstęp

Wprowadzenie

Cel prezentacji - przybliżenie procesu, problemów i ich rozwiązań związanych z budową solidnej bazy systemu operacyjnego, czyli dystrybucji GNU/Linux.

Tizen: Dystrybucja GNU/Linuksa

- **Dystrybucja GNU/Linuksa ze standardowymi komponentami:**
 - Jądro Linux
 - Narzędzia GNU
 - Xorg
- **... jak również trochę mniej standardowymi:**
 - Enlightenment
 - Wayland (opcja)
 - connman
 - ...

Tizen: Rozszerzenia

- **Messaging (SMS, MMS)**
- **Audio Policy (routing)**
- **PIM (Contacts, Calendar, Accounts)**
- **Sensors**
- **System settings**
- **Telephony services**
- **SIM management**

Tizen: Aplikacje i ich dystrybucja

- **Niestandardowy (dla GNU/Linuksa) pomysł na tworzenie i dystrybucję aplikacji:**
 - JavaScript (W3C APIs + specyficzne dla Tizena)
 - C++ (Bada API)
- **Dystrybucja aplikacji - "Tizen Store"**

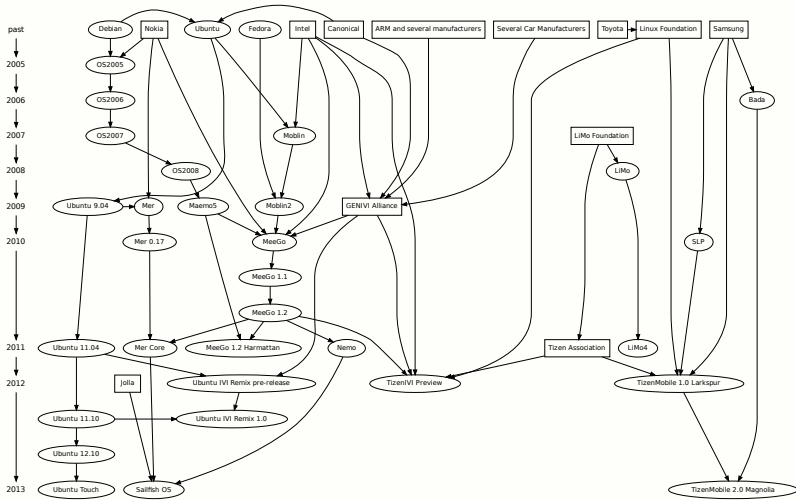
Tizen: Aplikacje i ich dystrybucja

- **Niestandardowy (dla GNU/Linuksa) pomysł na tworzenie i dystrybucję aplikacji:**
 - JavaScript (W3C APIs + specyficzne dla Tizena)
 - C++ (Bada API)
- **Dystrybucja aplikacji - "Tizen Store"**
- **Dystrybucja samych komponentów systemu - pakiety .rpm**



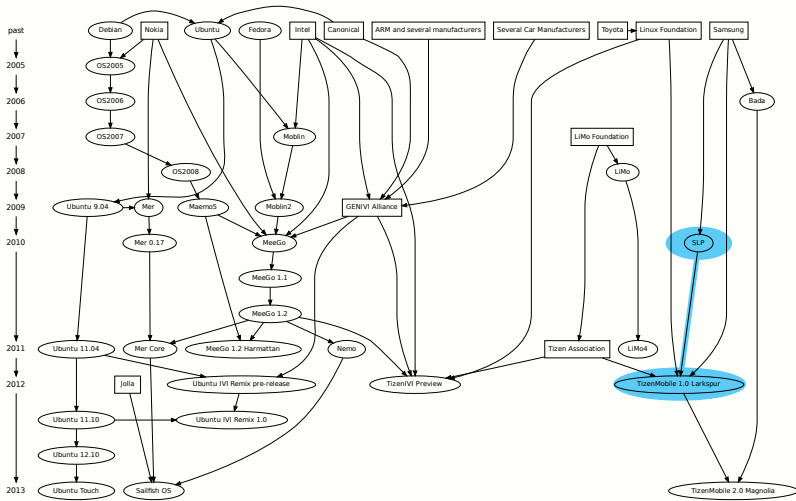
Historia

Historia <https://github.com/kumadasu/tizen-history>



Tizen History (In Construction March, 2013)

Historia <https://github.com/kumadasu/tizen-history>



Tizen History (In Construction March, 2013)

Samsung Linux Platform (SLP)

- **System operacyjny firmy Samsung oparty na GNU/Linuksie**
- **SLP zaczynał jako system wbudowany**

System wbudowany

Cechy systemu wbudowanego:

- **Realizuje jedną dobrze zdefiniowaną funkcję**
- **(Typowo) Preinstalowany**

System wbudowany

Cechy systemu wbudowanego:

- **Realizuje jedną dobrze zdefiniowaną funkcję**
- **(Typowo) Preinstalowany**
- **Bardzo mocno związany z produktem**
- **Produkt związany z datą wydania**

(Daty wydania nie zawsze są realne.)

System wbudowany (konsekwencje)

- "Optymalizowany" pod konkretny sprzęt (sleep 42)

System wbudowany (konsekwencje)

- "Optymalizowany" pod konkretny sprzęt (sleep 42)
- Kontrola dostępu nie zawsze traktowana z należytą uwagą (/dev/exynos-mem)

System wbudowany (konsekwencje)

- "Optymalizowany" pod konkretny sprzęt (sleep 42)
- Kontrola dostępu nie zawsze traktowana z należytą uwagą (/dev/exynos-mem)
- System bardzo okrojony (braki narzędzi lub dostępne ich zubożone wersje)

System wbudowany (konsekwencje)

- "Optymalizowany" pod konkretny sprzęt (sleep 42)
- Kontrola dostępu nie zawsze traktowana z należytą uwagą (/dev/exynos-mem)
- System bardzo okrojony (braki narzędzi lub dostępne ich zubożone wersje)
- Mnogość rozwiązań tymczasowych (pliki binarne w repozytorium)

SLP(2) circa 2010

- **scratchbox(1) do kompilacji skróśnej**
- **init(8) z busyboksa**
- **Większość procesów działa z uprawnieniami roota**
- **Pakiety .deb do dystrybucji oprogramowania**
- **W oryginalnym zamyśle "projektowany" na urządzenia smartphone**

Aspiracje (potencjalne zastosowania) SLP

- **System operacyjny dedykowany na specjalizowane systemy:**
 - Telefony
 - Tablety
 - TV
 - Aparaty fotograficzne
 - Systemy informacyjno-rozrywkowe ("infotainment")
 - ...

Aspiracje (potencjalne zastosowania) SLP

- **System operacyjny dedykowany na specjalizowane systemy:**
 - Telefony
 - Tablety
 - TV
 - Aparaty fotograficzne
 - Systemy informacyjno-rozrywkowe ("infotainment")
 - ...
- **Uniwersalny, otwarty system operacyjny na specjalizowane urządzenia dla produktów firmy Samsung (i innych, wedle uznania)**



Problem 1: Budowa oprogramowania

Cechy pożądanego środowiska budowania:

- Wygodna kompilacja skrośna istniejących projektów GNU/Linux (x86 -> ARM, amd64 -> x86)
- Środowisko przejrzyste dla programistów i wygodne w utrzymaniu
- Powtarzalne wyniki budowania (niezależne od środowiska programisty)

Cechy pożądanego środowiska budowania:

- Wygodna kompilacja skrośna istniejących projektów GNU/Linux (x86 -> ARM, amd64 -> x86)
- Środowisko przeźroczyste dla programistów i wygodne w utrzymaniu
- Powtarzalne wyniki budowania (niezależne od środowiska programisty)
- Niezależne środowiska dla różnych projektów, architektur sprzętowych, ...
- W pełni funkcjonalny system nie wymagający praw administratora

Wykorzystywane: scratchbox(1)

- **Bazuje na zmodyfikowanym środowisku, by stworzyć iluzję systemu docelowego:**
 - System docelowy (target) dostarcza biblioteki
 - Narzędzia (tools) dostarczają kompilator, linker, itp.
 - chroot(8) + bind mounty + symlinki + (magia) = "spójny system"

Wykorzystywane: scratchbox(1)

- **Bazuje na zmodyfikowanym środowisku, by stworzyć iluzję systemu docelowego:**
 - System docelowy (target) dostarcza biblioteki
 - Narzędzia (tools) dostarczają kompilator, linker, itp.
 - chroot(8) + bind mounty + symlinki + (magia) = "spójny system"
- **Narzędzia nie są w pełni funkcjonalnym systemem - trudna aktualizacja**
- **Środowisko programisty zupełnie inne niż budowania**
- **Instalacja globalna (/scratchbox) - wymaga praw administratora**
- **1 użytkownik = 1 środowisko budowania**

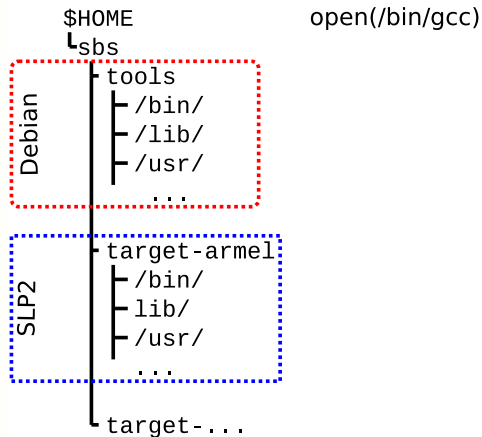
Alternatywy:

- **Natywna kompilacja**
- **Android - "make world"**
- **Debian/Ubuntu - multiarch**
- **Maemo/MeeGo - scratchbox2**

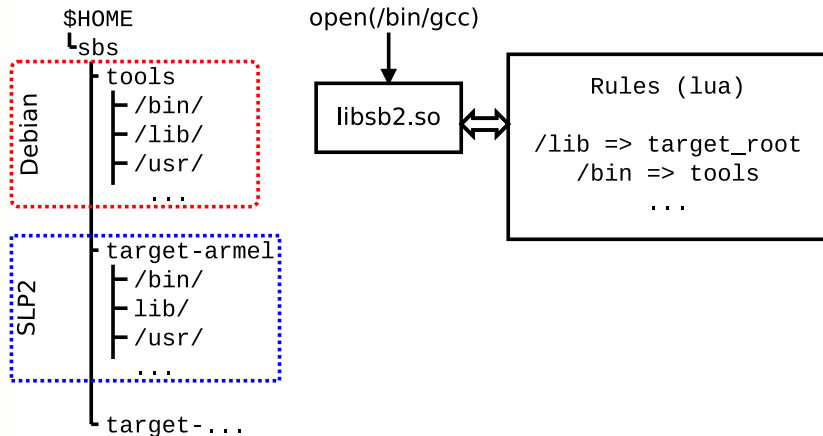
scratchbox 2

- Działa na zasadzie dynamicznego odwzorowywania ścieżek przy dostępie (access(2), open(2), ...)
- Reguły w języku Lua
- Bazuje na mechanizmie LD PRELOAD
- Nie wymaga praw administratora

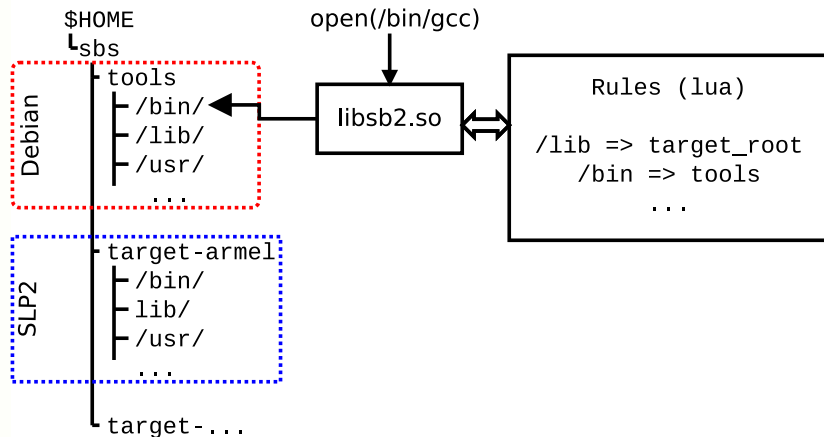
scratchbox 2 - przykład



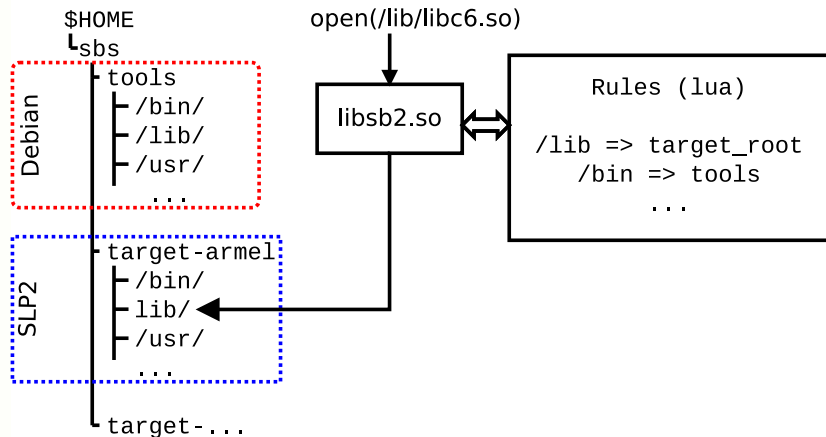
scratchbox 2 - przykład



scratchbox 2 - przykład



scratchbox 2 - przykład



scratchbox2 + samsung = sbs

- **sbs - SPRC/Samsung Build System**
 - Skrypt do stworzenia kompletnego środowiska (debootstrap, sb2-init, sb2)
 - Reguły specyficzne dla SLP
 - Przełączanie pomiędzy środowiskami

scratchbox2 + samsung = sbs

- **sbs - SPRC/Samsung Build System**
 - Skrypt do stworzenia kompletnego środowiska (debootstrap, sb2-init, sb2)
 - Reguły specyficzne dla SLP
 - Przełączanie pomiędzy środowiskami
- **scratchbox2**
 - Poprawki pozwalające na uruchamianie statycznych programów (qemu-native)

System budowania i zarządzanie kodem

- **openSUSE build service (OBS) - adresuje problemy, których nie rozwiązywał sbs**
 - Zaprojektowany z myślą o wymaganiach grupy SCM
 - Nastawiony na zarządzanie dużą ilością projektów
 - Scentralizowany i łatwo zarządzalny (web ui)

System budowania i zarządzanie kodem

- **openSUSE build service (OBS) - adresuje problemy, których nie rozwiązywał sbs**
 - Zaprojektowany z myślą o wymaganiach grupy SCM
 - Nastawiony na zarządzanie dużą ilością projektów
 - Scentralizowany i łatwo zarządzalny (web ui)

Zasada działania:

- **chroot(8), system docelowy + qemu-static-ARCH, zmodyfikowane pakiety i menedżer pakietów (instalacja z różnych architektur)**

System budowania i zarządzanie kodem

- **openSUSE build service (OBS) - adresuje problemy, których nie rozwiązywał sbs**
 - Zaprojektowany z myślą o wymaganiach grupy SCM
 - Nastawiony na zarządzanie dużą ilością projektów
 - Scentralizowany i łatwo zarządzalny (web ui)

Zasada działania:

- **chroot(8), system docelowy + qemu-static-ARCH, zmodyfikowane pakiety i menedżer pakietów (instalacja z różnych architektur)**
- **Gerrit do oceny jakości kodu**



Problem 2: `init(8)`

Stan zastany:

- **init(8) z busyboksa**
- **/etc/rc.d/rc.sysinit**
- **Skrypty serwisów - od 1 linii ("foo &") do 1xxx**
- **Synchronizacja uruchamiania usług:**
while [-e /tmp/foo]; do sleep 1; done && bar &

Stan zastany:

- **init(8) z busyboksa**
- **/etc/rc.d/rc.sysinit**
- **Skrypty serwisów - od 1 linii ("foo &") do 1xxx**
- **Synchronizacja uruchamiania usług:**
while [-e /tmp/foo]; do sleep 1; done && bar &
- **System "zoptymalizowany" - czasem działał**

ps -ef (składowe systemu)

- **Pojedyncze programy realizujące interfejs użytkownika (GUI)**

ps -ef (składowe systemu)

- **Pojedyncze programy realizujące interfejs użytkownika (GUI)**
- **Bardzo dużo usług klient/serwer (demonów) korzystających z różnorodnych mechanizmów IPC:**
 - Gniazda UNIX
 - D-Bus
 - SYSV IPC
 - vconf (pliki + inotify(2))

ps -ef (składowe systemu)

- **Pojedyncze programy realizujące interfejs użytkownika (GUI)**
- **Bardzo dużo usług klient/serwer (demonów) korzystających z różnorodnych mechanizmów IPC:**
 - Gniazda UNIX
 - D-Bus
 - SYSV IPC
 - vconf (pliki + inotify(2))
- **Serwisy restartujące krytyczne usługi i aplikacje:**
 - menu-daemon -> menu-screen

Alternatywne rozwiązania

- **sysvinit+insserv (tagi LSB)**
- **upstart**
- **systemd (v25)**

Alternatywne rozwiązania

- **sysvinit+insserv (tagi LSB)**
- **upstart**
- **systemd (v25)**
 - Deklaratywny opis systemu
 - Uruchamianie usług na żądanie (socket activation)
 - Uproszczenie zależności usług (dzięki powyższemu)
 - Domyślne zrównoleglanie uruchamianych usług
 - `systemd --user`

Wdrożenie systemd

Konsekwencje wdrożenia systemd:

Wdrożenie systemd

Konsekwencje wdrożenia systemd:

- **Brak zmian**

Wdrożenie systemd

Konsekwencje wdrożenia systemd:

- **Brak zmian**

Analiza działania systemu za pomocą narzędzi:

- **strace**
- **systemd-analyze**
- **bootgraph.pl (kernel)**
- **(systemd-)bootchart**

Wdrożenie systemd

Konsekwencje wdrożenia systemd:

- **Brak zmian**

Analiza działania systemu za pomocą narzędzi:

- **strace**
- **systemd-analyze**
- **bootgraph.pl (kernel)**
- **(systemd-)bootchart**

Symptomy problemów:

- **Niewykorzystane I/O, CPU**
- **Usługi uruchamiane sekwencyjnie**

Zbieranie dokładnych informacji - auditd

- **Podsystem audytu w Linuksie pozwala na bardzo dokładne śledzenie zachowania systemu.**

Zbieranie dokładnych informacji - auditd

- **Podsystem audytu w Linuksie pozwala na bardzo dokładne śledzenie zachowania systemu.**
- **IPC oznacza konieczność synchronizacji uruchamiania usług:**
 - Dostęp do plików/konfiguracji - open(2), write(2), inotify(2)
 - Usługi klient/serwer - connect(2), bind(2)

Zbieranie dokładnych informacji - auditd

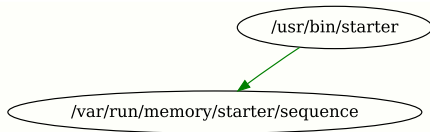
- **Podsystem audytu w Linuksie pozwala na bardzo dokładne śledzenie zachowania systemu.**
- **IPC oznacza konieczność synchronizacji uruchamiania usług:**
 - Dostęp do plików/konfiguracji - open(2), write(2), inotify(2)
 - Usługi klient/serwer - connect(2), bind(2)
- **Automatyczne generowanie grafów zależności (aureport + perl + graphviz/dot)**

auditd - przykład

/usr/bin/starter

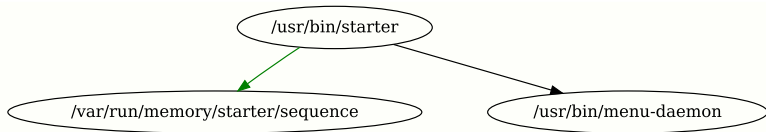
read(2), write(2), inotify(2), fork(2)

auditd - przykład



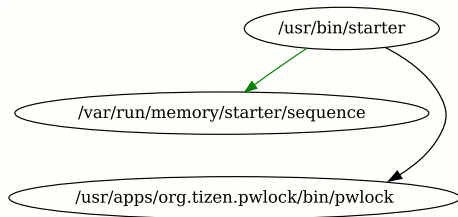
`read(2)`, `write(2)`, `inotify(2)`, `fork(2)`

auditd - przykład



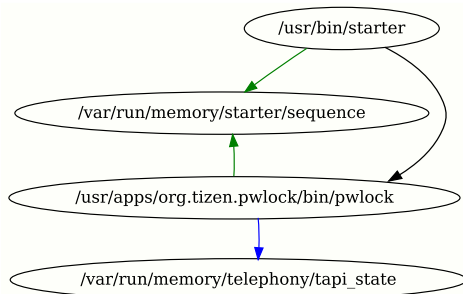
`read(2), write(2), inotify(2), fork(2)`

auditd - przykład



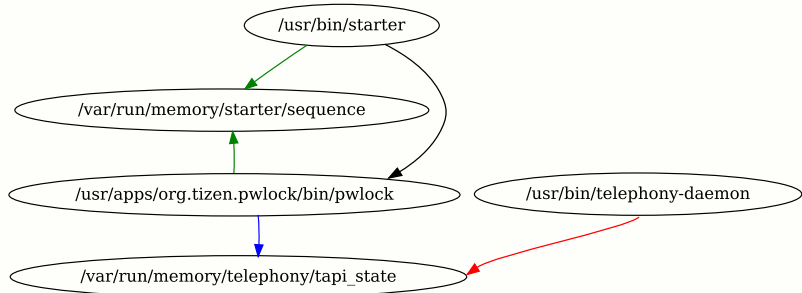
`read(2)`, `write(2)`, `inotify(2)`, `fork(2)`

auditd - przykład



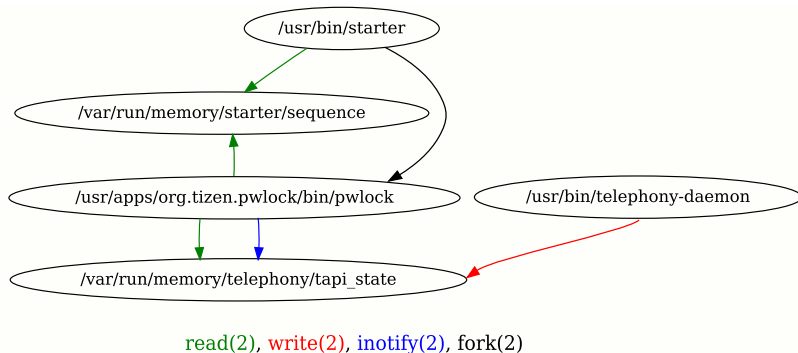
`read(2), write(2), inotify(2), fork(2)`

auditd - przykład

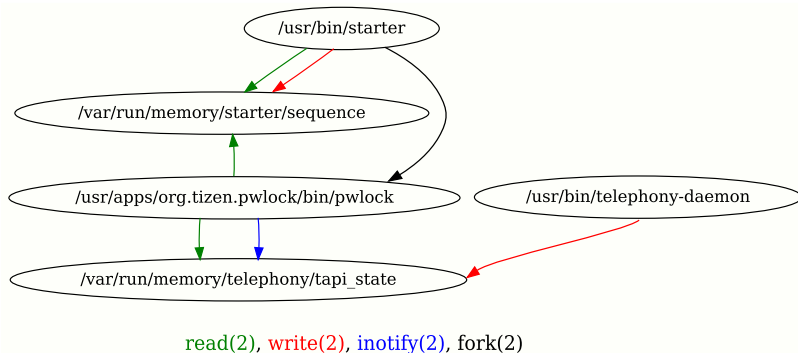


read(2), write(2), inotify(2), fork(2)

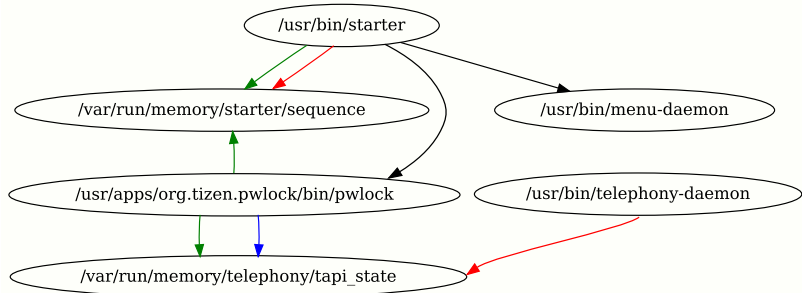
auditd - przykład



auditd - przykład



auditd - przykład



read(2), write(2), inotify(2), fork(2)

auditd

- **Dodatkowo śledzenie:**
 - `sync(2)`, `f*sync(2)`
 - `execve(3)`
 - ...

init(8) dziś i jutro

- **systemd v204 + natywna konfiguracja**
- **Wiele serwisów uruchamianych na żądanie, wiele rozważanych (Xorg)**



Problem 3: Niezawodność systemu

Stan zastany:

- Większość programów działa z prawami administratora (w tym window manager)
- Zatrzymanie procesu często kończy się wymuszonym restarem systemu (watchdog)

Stan zastany:

- **Większość programów działa z prawami administratora (w tym window manager)**
- **Zatrzymanie procesu często kończy się wymuszonym restarem systemu (watchdog)**

W konsekwencji:

- **Niemożliwe do zrealizowania jakiegokolwiek security**

Stan zastany:

- **Większość programów działa z prawami administratora (w tym window manager)**
- **Zatrzymanie procesu często kończy się wymuszonym restarem systemu (watchdog)**

W konsekwencji:

- **Niemożliwe do zrealizowania jakiegokolwiek security**

Systemy Uniksowe dostarczyły podstawowego rozwiązania zagadnienia security ponad 40 lat temu - użytkownicy, grupy (tzw. DAC).

Zarządzanie sesją użytkownika

- **Programy sesji użytkownika mają podobne wymagania jak systemowe:**
 - Zarządzanie cyklem życia (w tym automatyczny restart)
 - Uruchamianie usług na żądanie
 - Monitorowanie

Zarządzanie sesją użytkownika

- **Programy sesji użytkownika mają podobne wymagania jak systemowe:**
 - Zarządzanie cyklem życia (w tym automatyczny restart)
 - Uruchamianie usług na żądanie
 - Monitorowanie
- **systemd --user**
 - Sesja graficzna (xorg-launch-helper)
 - Sesyjny D-Bus uruchamiany na żądanie

Niezawodność dziś i jutro

- Usługi nieuprzywilejowane w osobnej sesji
- SMACK do drobnoziarnistej kontroli dostępu



Dalsze kroki

Quo vadis TizenOS?

- **Więcej GNU/Linuksa w Tizenie**

Quo vadis TizenOS?

- **Więcej GNU/Linuksa w Tizenie**
- **Stworzenie nowej dystrybucji GNU/Linuksa**
- **Naprawienie błędów**
- **???**
- **PROFIT!!!**



Q&A

Pytania i odpowiedzi

Autor niniejszej prezentacji nie posiada informacji czy i kiedy będzie wydany wykorzystujący Tizena:

- **telefon**
- **telewizor**
- **samochód**
- **czołg(?)**

:)

Dziękuję za uwagę

Karol Lewandowski <k.lewandowsk@samsung.com>
lmctl @freenode (#tizen)

Linki

- **sbs:** <https://review.tizen.org/git/?p=tools/sbs.git;a=summary>
- **Gerrit:** <http://review.tizen.org/gerrit>
- **Grafy auditd:** https://wiki.tizen.org/wiki/System/Dependency_graphs
- **Lista dyskusyjna:**
<https://lists.tizen.org/pipermail/dev/>