jesień linuksowa | jesien.org

11-13 Październik 2013 - Szczyrk, Polska

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

Karol Lewandowski

Samsung R&D Institute Poland

Outline

- Ta prezentacja nie jest o…
- Czym jest Tizen?
- Archeologia
- Samsung Linux Platform
- Problem 1: Budowa oprogramowania
- Problem 2: init(8)
- Problem 3: Niezawodność systemu
- Co dalej?
- Q&A

Ta prezentacja nie jest o…

… produktach

"Kiedy będzie wykorzystujący Tizena:

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

Więc?

Budowanie solidnej bazy systemu operacyjnego, czyli dystrybucji GNU/Linuksa.

Czym jest Tizen?

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

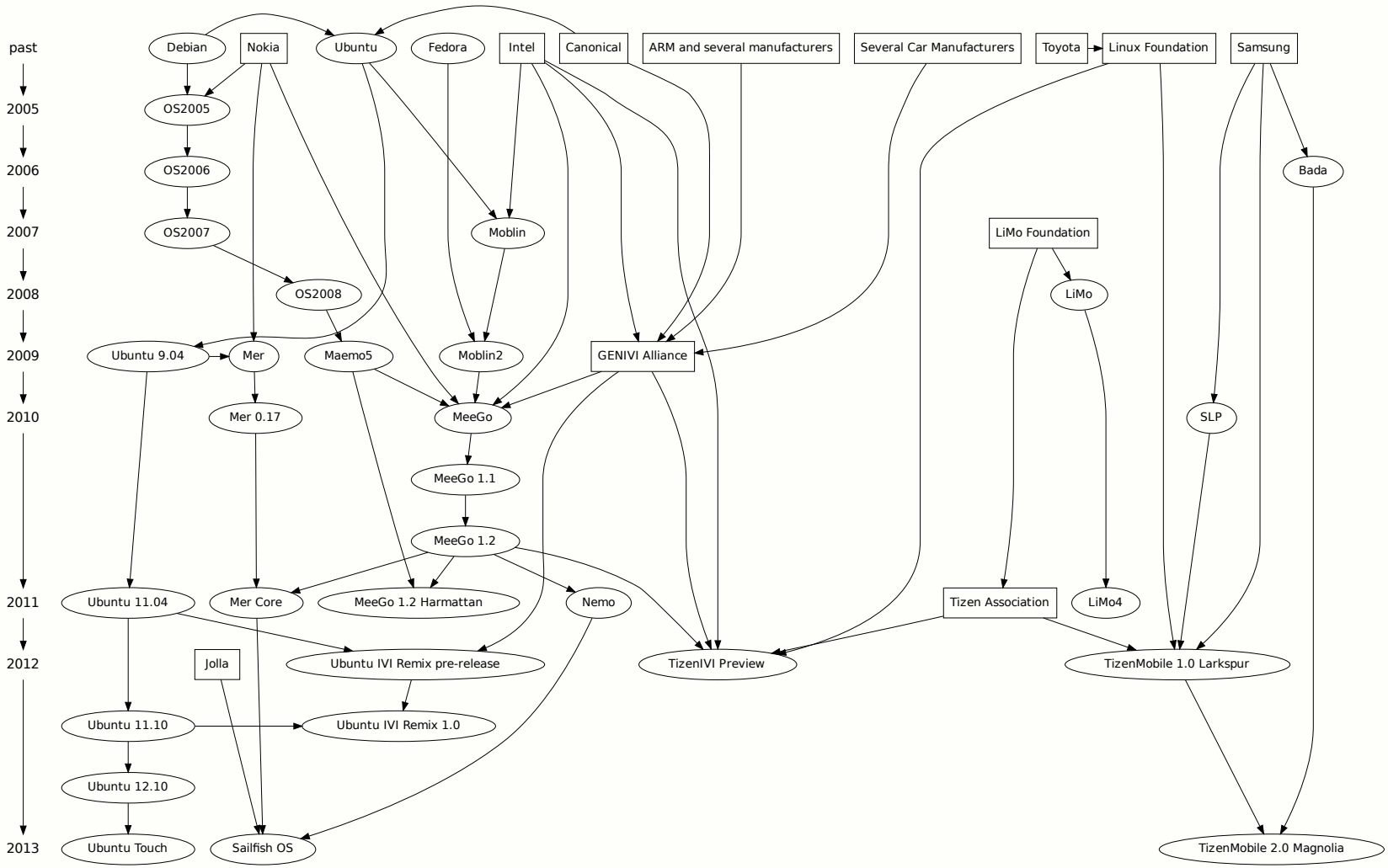
- **Messaging (SMS, MMS, Email)**
- **Audio Policy (routing)**
- **PIM (Contacts, Calendar, Accounts)**
- **Sensors**
- **System settings**
- **Telephony services**
- **SIM management**

Tizen: Dystrybucja oprogramowania

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

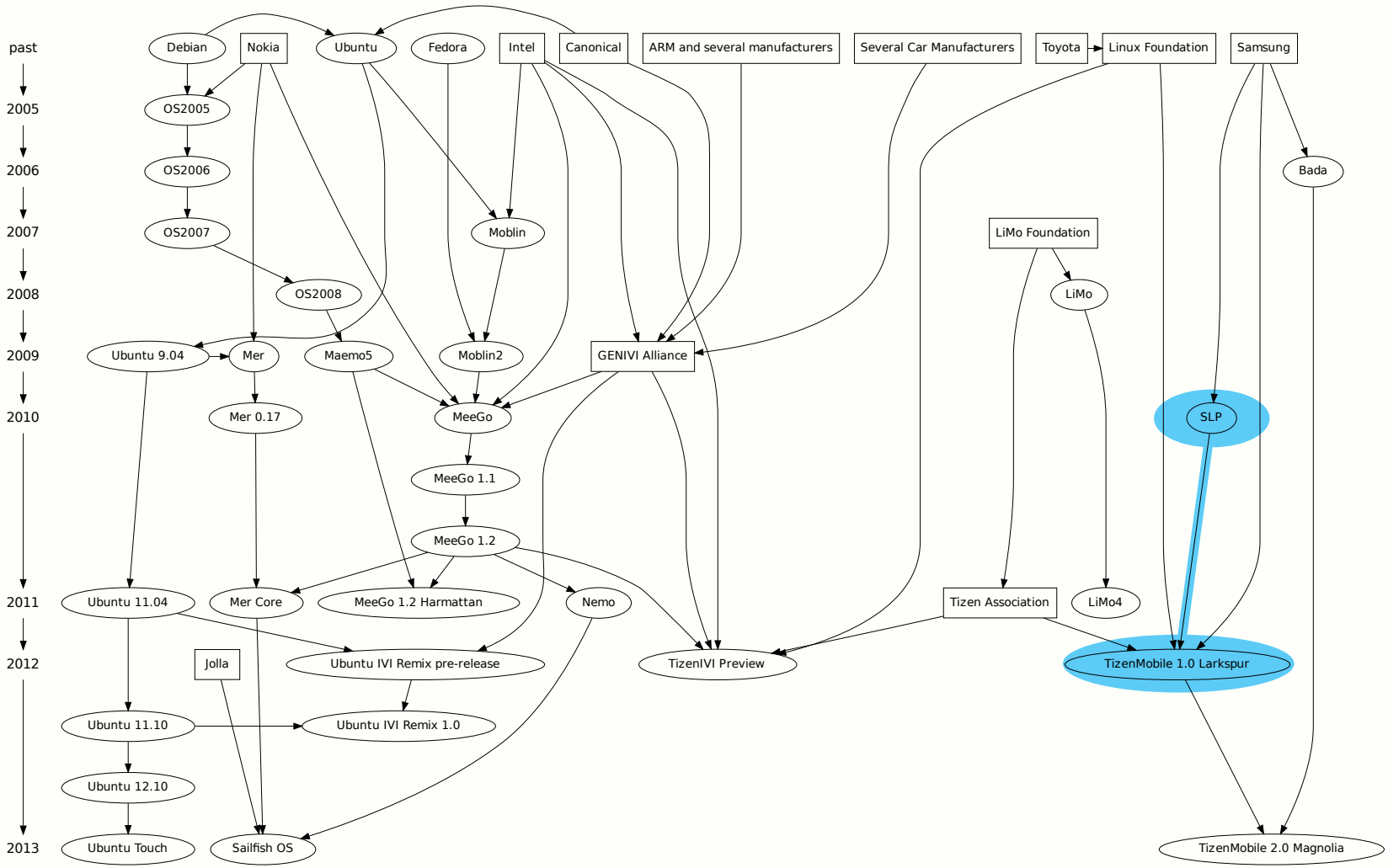
Archeologia

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

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 skrośnej**
- **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 przeźroczyste 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 funcjonalny 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**

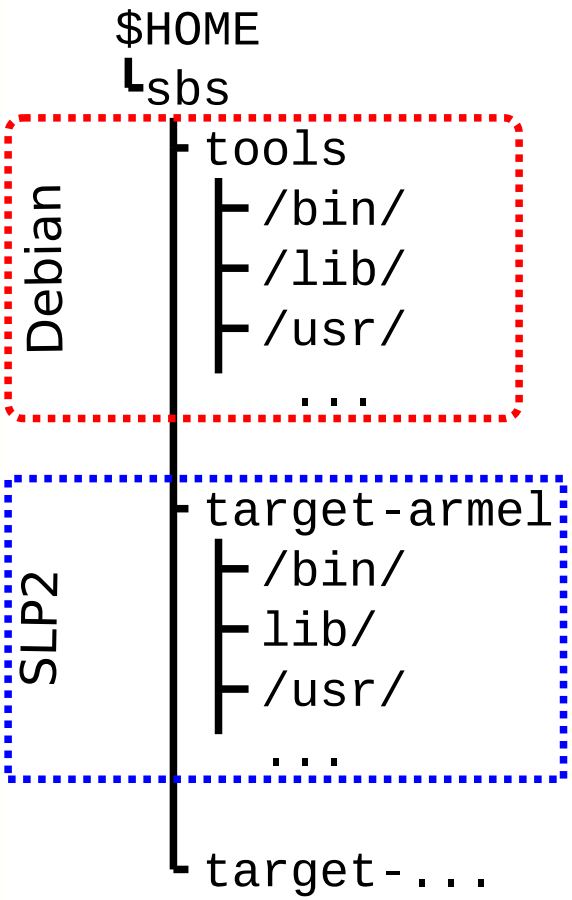
Alternatywy:

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

scratchbox 2

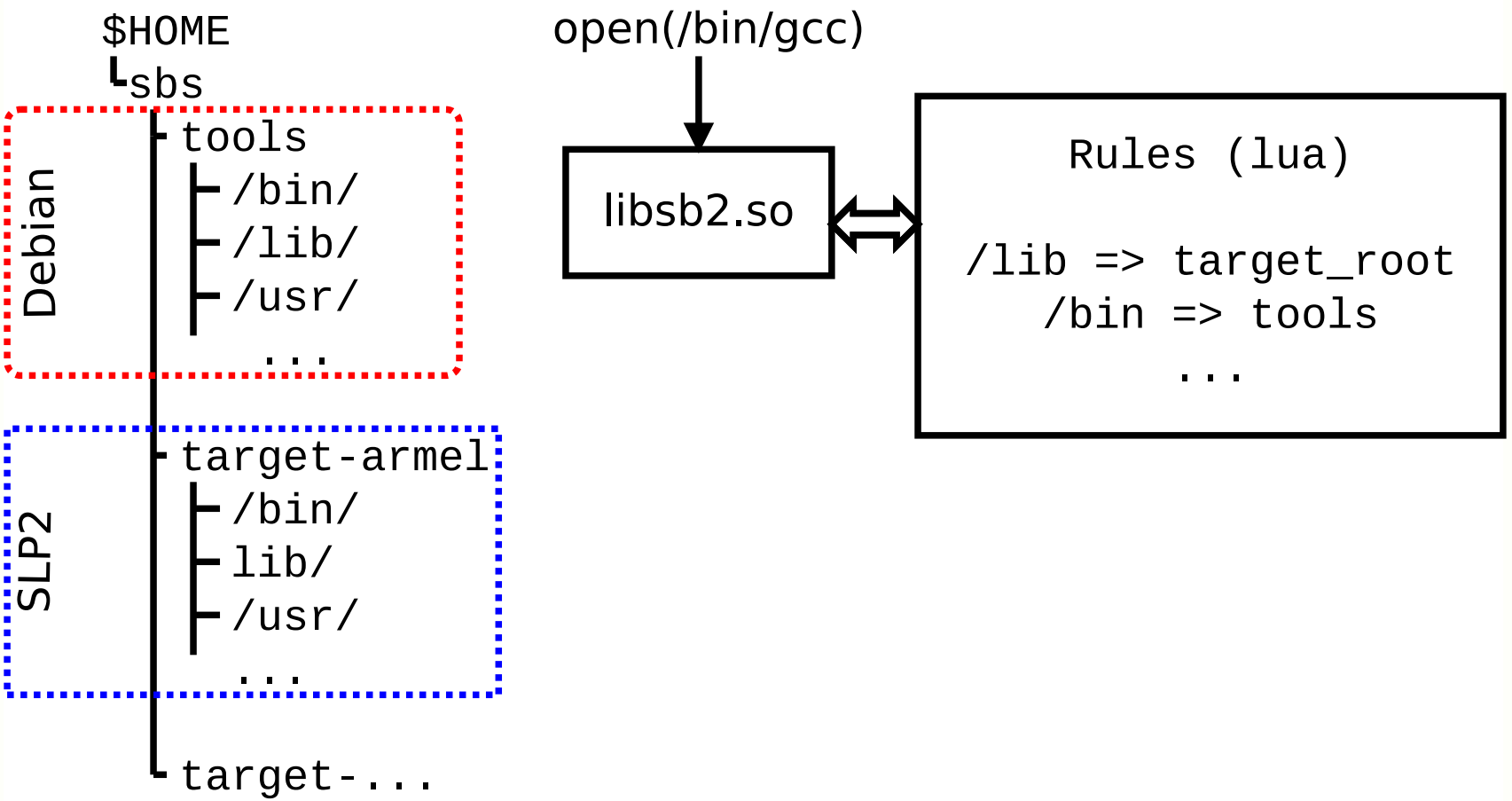
- **Działa na zasadzie dynamicznego odwzorowywania ścieżek przy dostępie (access(2), open(2), …)**
- **Reguły w języku Lua**
- **Bazuje na mechanizmie LD PRELOAD**
- **Nie wymaga praw administratora**

scratchbox 2 - przykład

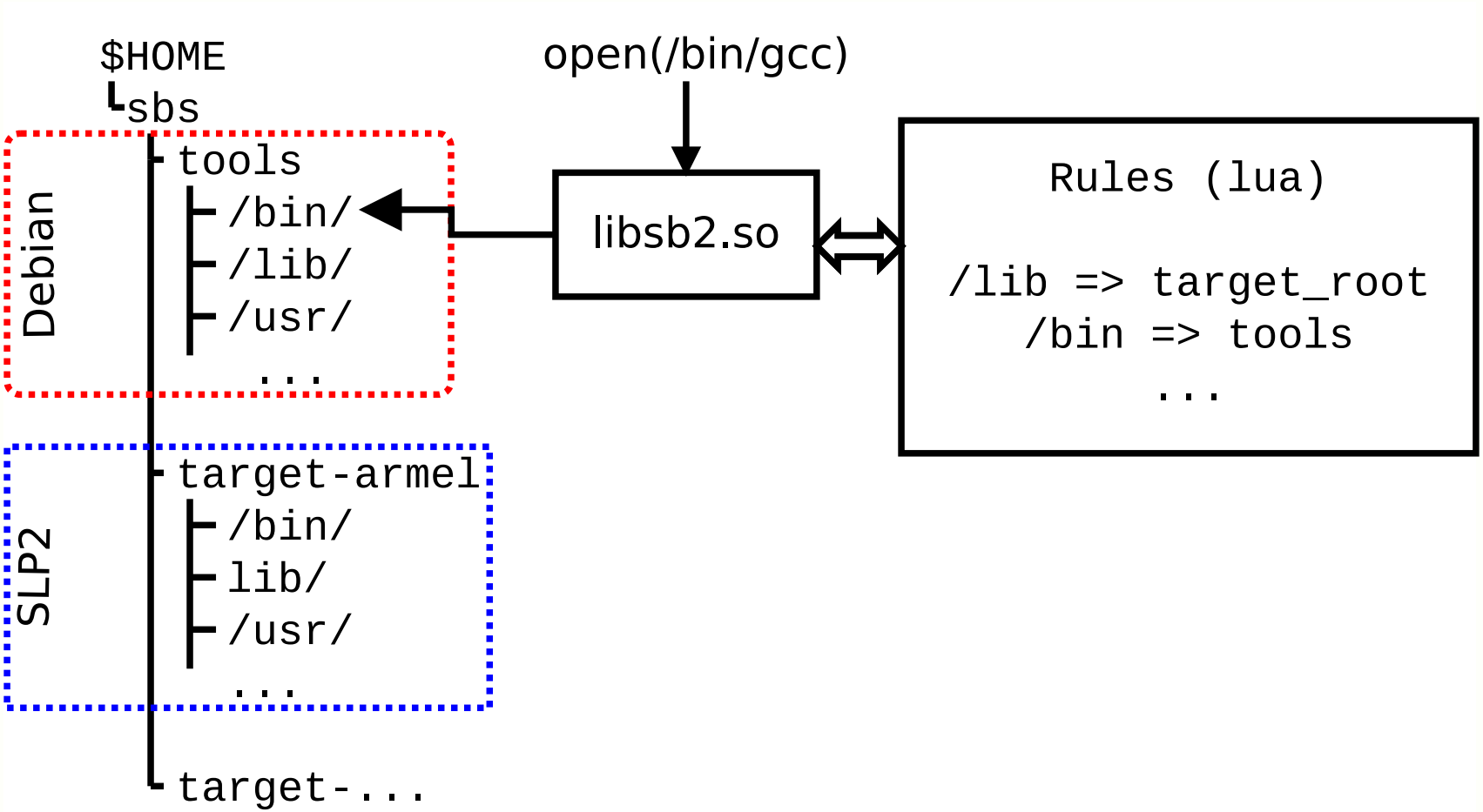


open(/bin/gcc)

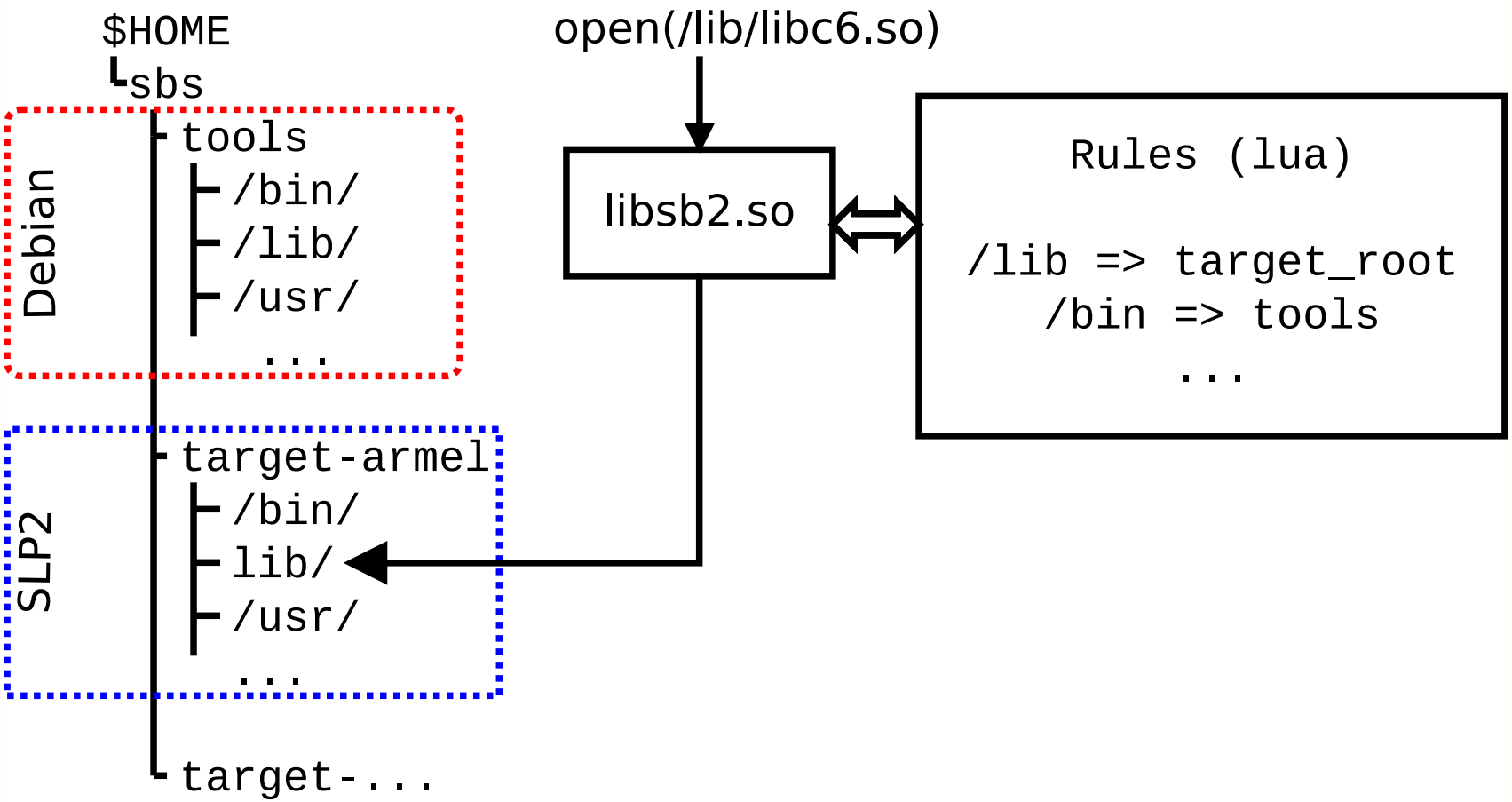
scratchbox 2 - przykład



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)

Dzisiejszy system budowania w Tizenie - OBS/gbs

- **Scentralizowany system budowania wykorzystywany w openSUSE**
- **chroot(8), system docelowy + qemu-static-ARCH, zmodyfikowane pakiety i menedżer pakietów (instalacja z różnych architektur)**
- **Narzędzia**
  - gbs(1) - git build system
  - osc(1) - openSUSE build service cli

Dzisiejszy system budowania w Tizenie - OBS/gbs

- **Scentralizowany system budowania wykorzystywany w openSUSE**
- **chroot(8), system docelowy + qemu-static-ARCH, zmodyfikowane pakiety i menedżer pakietów (instalacja z różnych architektur)**
- **Narzędzia**
  - gbs(1) - git build system
  - osc(1) - openSUSE build service cli
- **Wymaga uprawnień admistratora**
- **.deb -> .rpm**

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)

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

ps -ef (składowe systemu)

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

ps -ef (składowe systemu)

- **Pojedyńcze programy realizujące interfejs użytkownika (GUI)**
- **Bardzo dużo usług klient/serwer (demonów) korzystających z różnorakich 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**

Alternatywne rozwiązania

- **sysvinit+insserv (tagi LSB)**
- **upstart**
- **systemd**
  - niekompatybilny, deklaratywny(!) opis systemu
  - uruchamianie usług na żądanie (socket activation)
  - uproszczenie zależności usług (dzięki powyższemu)
  - domyślne zrównoleglanie uruchamianych usług
  - systemd --user

systemd (konsekwencje)

systemd (konsekwencje)

Brak zmian.

systemd (konsekwencje)

Brak zmian.

Dlaczego?

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

Symptomy:

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

auditd

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

auditd

- **Podsystem audytu w Linuksie pozwala na bardzo dokładne śledzenie zachowania systemu.**
- **IPC oznacza konieczność synchronizacji uruchamiania usług:**
  - open(2), write(2), inotify(2)
  - connect(2), bind(2)

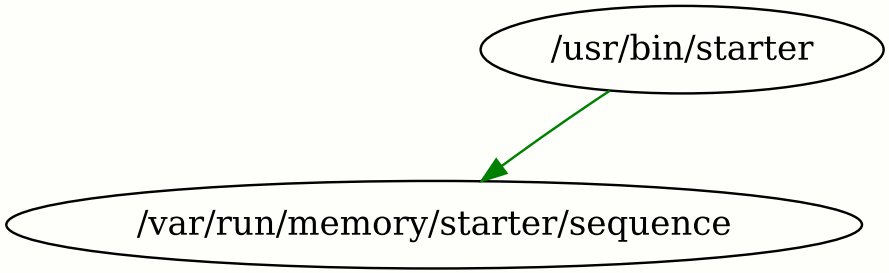
auditd

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

auditd - przykład

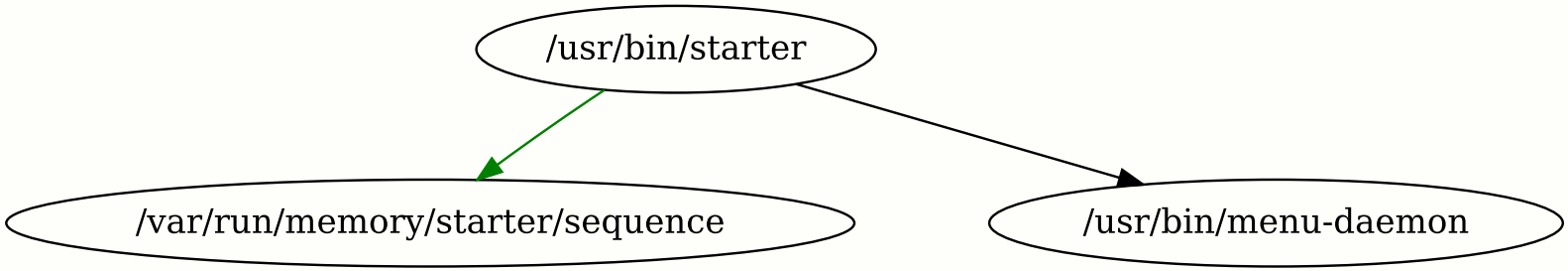
read(2), write(2), inotify(2), fork(2)

auditd - przykład



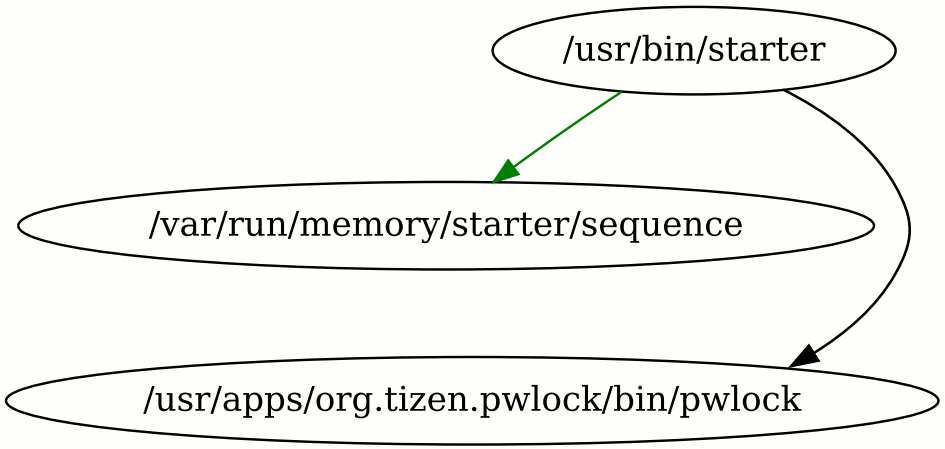
read(2), write(2), inotify(2), fork(2)

auditd - przykład



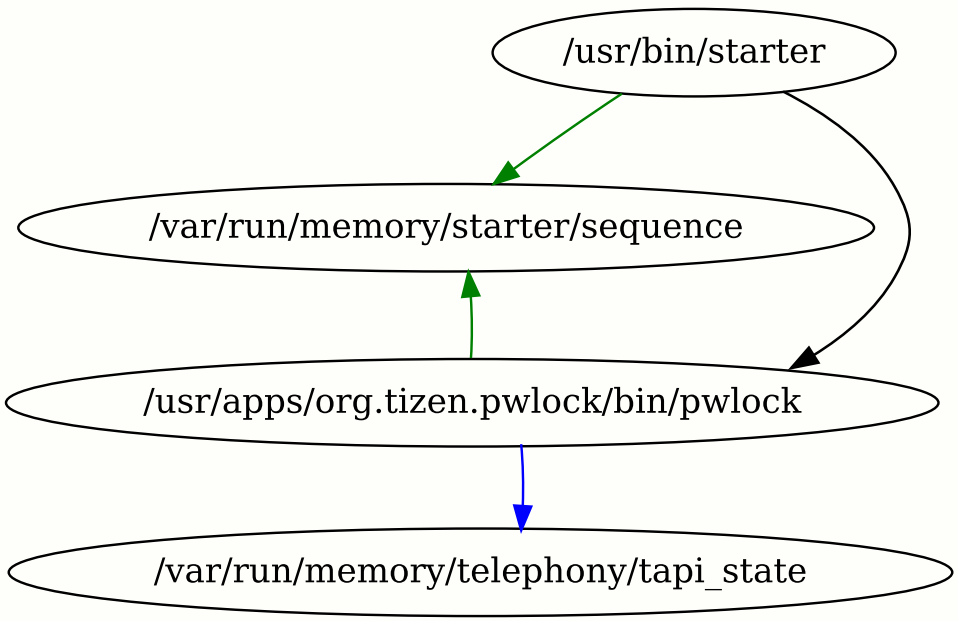
read(2), write(2), inotify(2), fork(2)

auditd - przykład



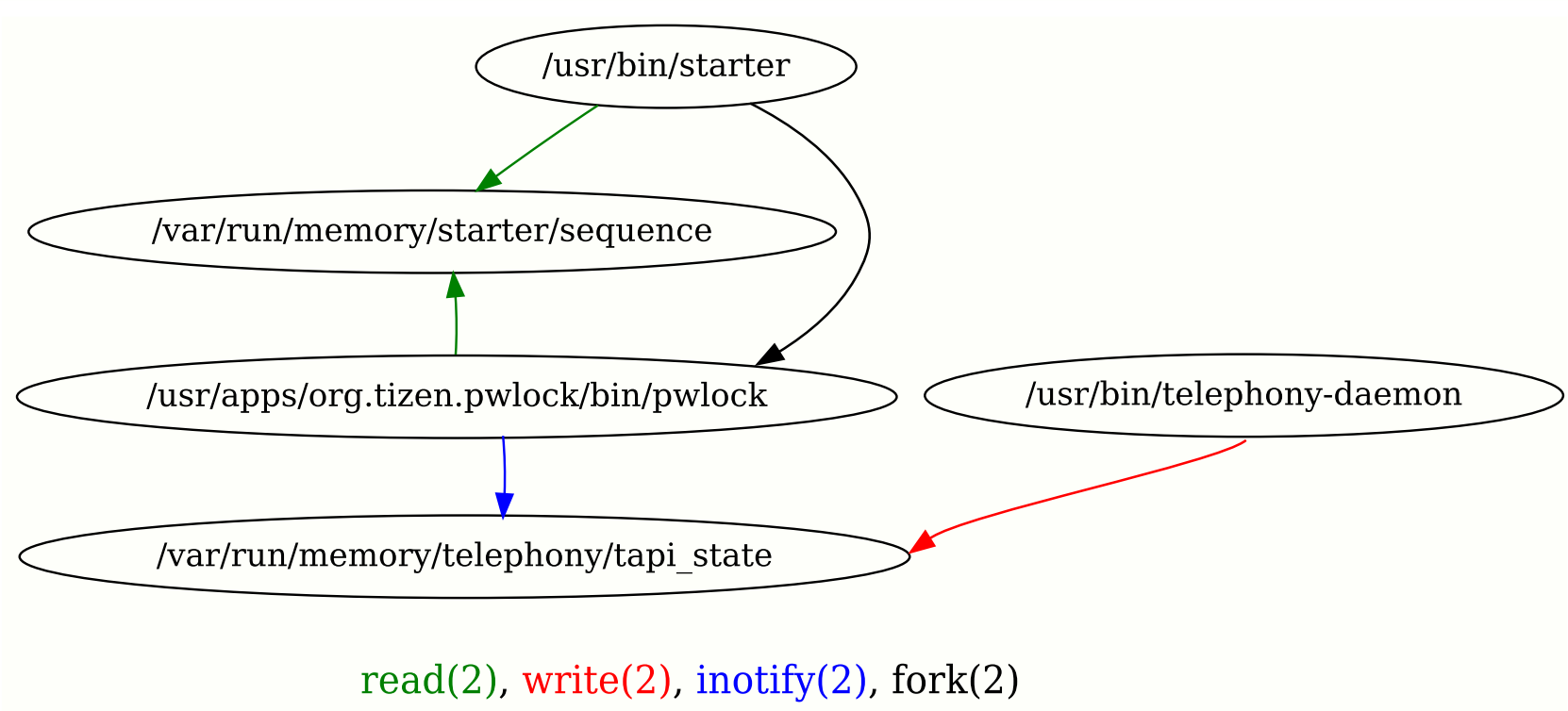
read(2), write(2), inotify(2), fork(2)

auditd - przykład

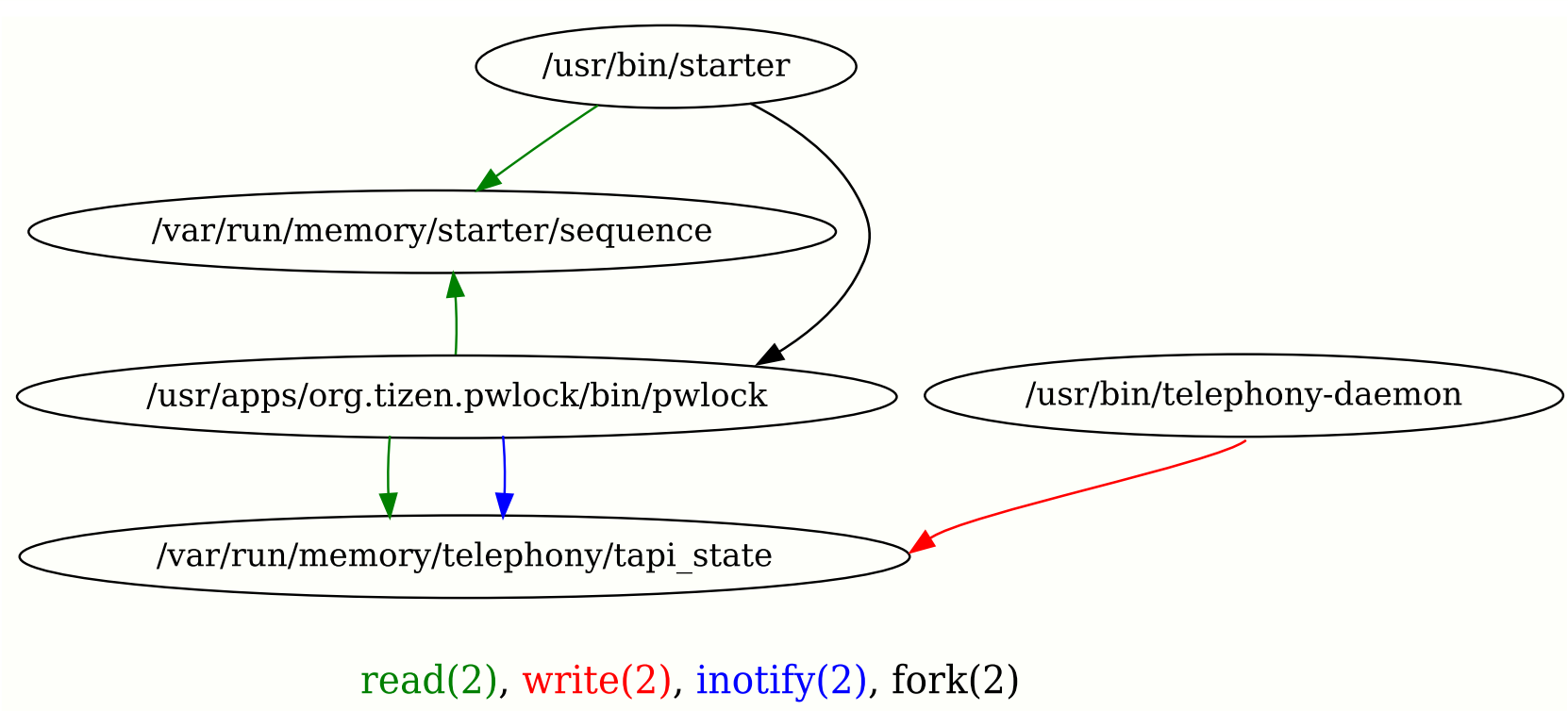


read(2), write(2), inotify(2), fork(2)

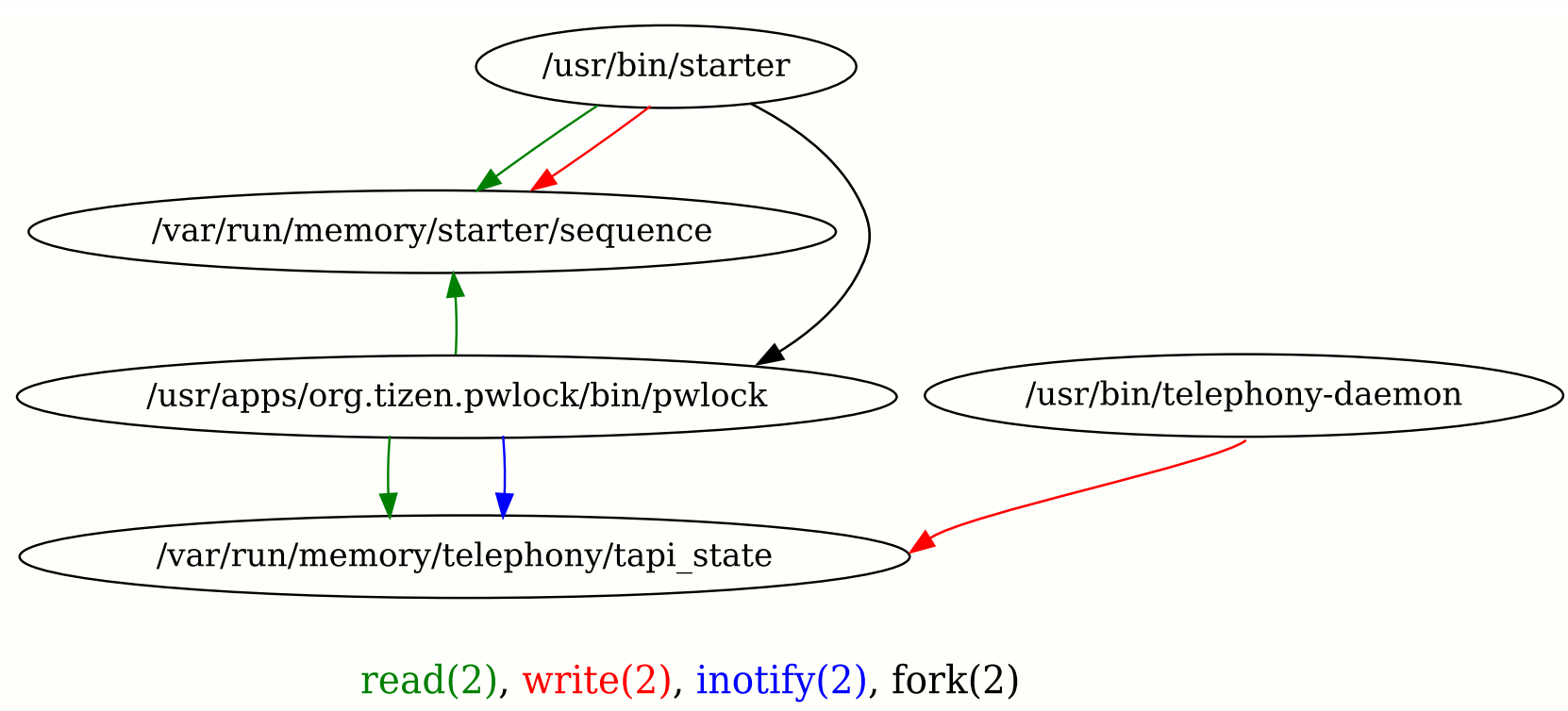
auditd - przykład



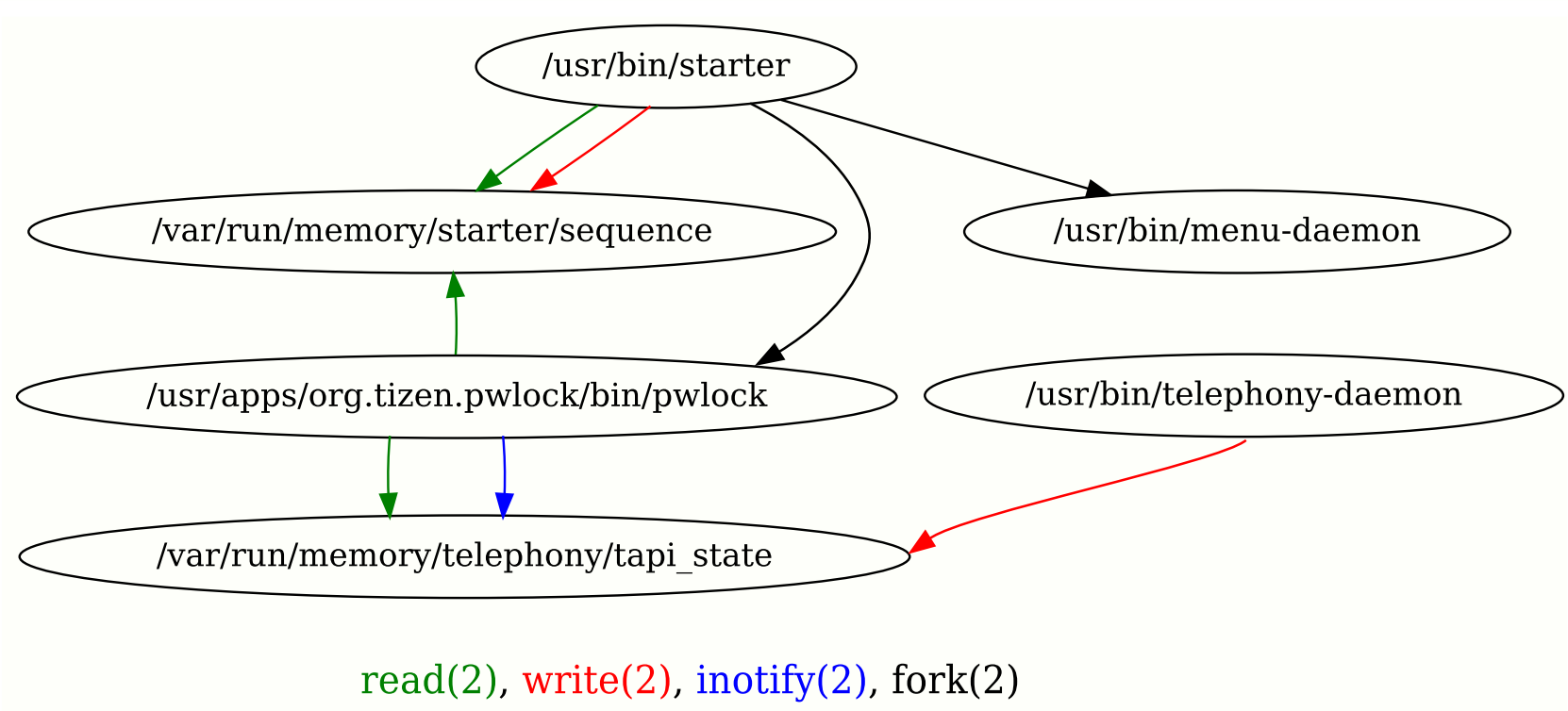
auditd - przykład



auditd - przykład



auditd - przykład



auditd

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

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 jakiekolwiek 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 jakiekolwiek 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

Co dalej?

Quo vadis TizenOS?

- **Więcej GNU/Linuksa w Tizenie (udisks, …)**
- **{Xorg, …} uruchamiany na żądanie**
- **Multi user, multi head, multi seat**
- **…**

Q&A

Dziękuję za uwagę

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