Tema 2 - Christmas Tree

Obiective

- construirea unui dispozitiv embedded complet
- lucrul cu Yocto, generarea de distribuții pentru sisteme embedded
- configurarea Yocto și adăugarea de pachete necesare unei aplicații concrete
- configurări de bază pentru lucrul în rețea
- scripting și folosirea utilitarelor dintr-un sistem Linux
- optimizări de spațiu folosit (opțional)

Background

Presupunem că avem un sistem embedded Linux conectat la un display multicolor de dimensiuni irelevante și o interfață web. Display-ul este capabil să afișeze brazi împodobiți pe baza unui text ASCII. Textul ASCII se încarcă prin interfața web, este prelucrat de către sistemul embedded, iar apoi este afișat pe display.

Display-ul este conectat la sistemul embedded printr-o interfață serială. În condițiile temei, considerăm că display-ul reprezintă fereastra QEMU.

Enunț

Pentru tema 2 se dorește realizarea unui sistem embedded bazat pe placa *Versatile Platform Baseboard* capabil să funcționeze ca un controller de afișaj multicolor conectat la Internet (IoT). Sistemul va fi construit cu ajutorul Yocto și va rula în QEMU. El va trebui să expună o interfață web care să permită încărcarea textelor ASCII și să afișeze în format multicolor arta ASCII.

Cerințe

- Basic:
  - Imaginea sistemului trebuie generată cu Yocto, pornind de la instrucțiunile prezente în resurse.
  - Sistemul trebuie să ruleze în QEMU, folosind kernel [https://drive.google.com/file/d/0B0lgiPZNMMyvMWEtZS1zME9lQjQ]-ul special compilat
  - Sistemul trebuie să aibă un utilizator `root` cu parola `labsi`
  - Sistemul trebuie să își configureze automat IP-ul folosind DHCP
  - Sistemul trebuie să ruleze SSH pe portul 22
  - Sistemul trebuie să ruleze daemon-ul Avahi/mDNS și să răspundă la numele `tema2.local`
- Control
  - Display-ul se găsește pe interfața serială `tty0`, care va fi configurată să trimită datele la stdout.
  - Sistemul embedded trebuie să conțină o interfață web prin care sa accepte încărcarea fișierelor care conțin o artă ASCII.
  - Sistemul poate să prelucreze în mod dinamic o artă ASCII de îndată ce aceasta este încărcată și să o afișeze în consola.
  - Sistemul poate să substituie corect caracterele ASCII în culori:
    - Spațiul dintre două caractere / și \ va avea culoarea verde. Inclusiv caracterele / și \ se vor substitui în culoarea verde.

- Spațiul dintre două caractere | va avea culoarea negru. Inclusiv caracterul | se va substitui în culoarea negru.
- Spațiul dintre două caractere ^ va avea culoarea galben. Inclusiv caracterul ^ se va substitui în culoarea galben.
- Caracterul ~ se va substitui în culoarea roșu.
- Caracterul ` se va substitui în culoarea albastru.
- Caracterul * se va substitui în culoarea magenta.
- Caracterul & se va substitui în culoarea alb.
- Caracterul + se va substitui în culoarea cyan.

Exemplu de brad complet în format ASCII: brad_complet.

Vă puteți folosi și de următoarele exemple de brazi: brad_1, brad_2, brad_3.

Pentru "colorarea" terminalului, vă puteți folosi de API-ul pus la dispoziție de biblioteca ncurses [https://tldp.org/HOWTO/NCURSES-Programming-HOWTO] sau direct de utilitarul tput [https://www.tldp.org/HOWTO/Bash-Prompt-HOWTO/x405.html].

Tema va fi testată în QEMU (v2.0+), pentru placa `versatilepb`.

Pentru ușurința navigării pachetelor puteți folosi interfața grafică *Toaster* pentru a construi imaginea. Puteți pleca de la imaginea de bază pentru RaspberryPi, `rpi-basic-image`.

Trimitere

Tema va fi trimisă în două componente:

- arhiva propriu-zisă → pe orice platformă de hosting doriți (ex: Dropbox, Google Drive, transfer.ro etc.)
  - link-ul trebuie să fie valid minim o săptămână din momentul expirării deadline-ului hard
- metainformații → pe Moodle

Arhiva trebuie să conțină:

- partiția ext3 a sistemului (sau de alt tip, dacă aveți motive solide)
- imaginea kernel-ului pentru QEMU (o puteți folosi pe cea din laborator)
- script de pornire QEMU - `launch.sh`
- sursele aplicației/script-urile folosite pentru controlul display-ului și pentru interfața web
- fișier `README` cu explicații referitoare la funcționarea soluției, opțiuni speciale de configurare/optimizare folosite etc.

Metainformațiile constau dintr-un fișier care să conțină:

- link către arhiva **.zip** cu rezolvarea propriu-zisă
- hash-ul md5 al arhivei (obținut cu `md5sum`)

Nu vor fi punctate temele care nu au hash-ul MD5 al arhivei încărcat pe Moodle sau cele al căror hash nu corespunde cu arhiva downloadată de pe platforma de hosting la momentul corectării.

Notare

Din 100p total, aveți:

- (20p) Imaginea și funcționalitățile de bază.

- (25p) Interfață web din care se poate încărca un text ASCII.
- (40p) Implementarea translatării din text ASCII în culori afișate în fereastra QEMU.
- (15p) Readme scris clar și care descrie complet arhitectura și implementarea funcționalităților cerute

Bonus:

- (5p) Optimizări deosebite de spațiu ale imaginii finale (cele mai mici 5 imagini trimise, care implementează toate funcționalitățile)
- (5-10p) Wow factor (extra funcționalități, design plăcut, eleganță în implementare etc.)
- (5p) Salvarea unui log cu acțiunile efectuate de aplicație și limitarea spațiului consumat folosind *logrotate*

Precizări

- Tema are doar deadline hard.
- Compilarea unei imagini cu Yocto necesită 20GB+
- Prima compilare poate dura intre 1 si 4 ore în funcție de performanța calculatorului (procesor + I/O) și viteza conexiunii la Internet, timp în care sunt downloadate și compilate sursele pachetelor incluse în imagine
- Compilările ulterioare vor refolosi majoritatea pachetelor generate și vor dura mult mai puțin
- Imaginea generată (inclusiv cea de bază), precum și restul temei, este individuală; nu o share-uiți cu colegii, deoarece vom verifica acest lucru

Resurse

- Readme Yocto
- Kernel pentru QEMU [https://drive.google.com/file/d/0B0lgiPZNMMyvMWEtZS1zME9lQjQ]
- Exemplu de brad complet

si/lab/2020/tema2.txt · Last modified: 2020/12/24 23:19 by dragos_florin.costea