# 9. Tétel

Ismertesse a Yocto Project fejlesztői keretrendszer funkcióját, felépítését, működését!

## Keresztplatformos beágyazott rendszer készítése

* ha a rendszer elkészítéséhez használt és a cél platform jelentősen eltér egymástól 🡪 és általában így van
* keresztfordítós megoldás akkor is jó, ha azonos a két platform: az alkalmazások egy „virtuális” környezetben nőjjenek fel (csak azt használják, amit megadunk nekik, így nem matatva a rendszerben)
* **szükséges cuccok:**
  + platformfüggő kernel-header állományok
  + binutils csomag lefordítása
  + C fordító (pl. gcc-vel glibc fordítása 🡪 glibc-vel gcc teljes lefordítása)
  + fejlesztői könyvtárak fordítása
* **automatizált eszközök:**
  + ***Buildroot***
    - makefile-ok és patch-ek gyűjteménye (uClibc-t használ)
    - kicsit nehézkes
  + ***OpenEmbedded***
    - BitBake receptek: fordítási műveletek + csomagok előállításának módjai
    - számos architektúra támogatsása
    - könnyen bővíthető
  + ***YoctoProject***
    - OpenEmbedded alapú
    - egyszerűsíti a beágyazott rendszer készítést, könnyíti a fejlesztést különböző hardverekre
    - jól dokumentált
    - Eclipse-be integrálódik
    - szimulációs környezet és SDK-t is lehet vele készíteni
  + ***Scratbox***
    - OS emulálás

## Yocto Project felépítése

* bitbake – a program, ami a recepteket tartalmazza
* build – a munkakönyvtár
* documentation
* Rétegek: meta, meta-yocto, meta-yocto-bsp
* classes – BBClass állományok, amelyek receptekben felhasználható műveleteket implementálnak
* conf – Konfigurációs állományok
  + - bitbake.conf – BB fő konfigja
    - distro - Disztribúciós
    - machine – Architektúrák
* files – Egyéb állományok
* recipes
  + - Receptek a különböző SW verziókhoz
    - Verzió független állományok
    - Verziófüggő állományok (patch, config)
* site – OS függő beállítások
* scripts – automatizáló szkriptek

### Saját rétegek: yocto-bsp create <bsp-name> <krach>

* új réteg neve: bsp-name
* krach – kernel architektúra
* BSP jellegű rétegek, de ebből később törölhetőek a HW konfig részek, vagy asjátot is létrehozhatunk
* conf/layer.conf – réteg beállításai
  + réteg neve
  + receptek elérhetősége
  + réteg prioritása
* új architektúra konfig: conf/machine/
* saját receptek: recipes-\*/

## Yocto Valósidejű rendszer készítésére

* csak a kernel más + néhány SW eszközt hozzáadhatunk
* linux-yocto-repository repo:
  + itt találhatóak preempt-rt ágakat 🡪 ezek megfelelő felhasználásával legenerálható a kernel 🡪recept kell
  + sok architektúrához már készítettek receptet 🡪 meg kell keresni a Yocto rendszerébe

## Beágyazott rendszer telepítése

* **Build eredménye:**
  + deploy könyvtárba ipk, rpm, deb formátumú csomagokat generál
  + ha egyébb fejlesztői eszközöket is előállítottunk, azok csomagjai az sdk könyvtárba kerülnek
* **Telepítés A:**
  + háttértár partícionálása (általában 1 partíció)
  + állományrendszer elkészítése a partíción
  + partíció felcsatolása
  + a buildelt csomagot ki kell tömöríteni a partícióra
  + egyéb konfig, ha szükséges
* **Telepítés B:** egyes architektúrákra már készen van komplett lemezképben a rendszer
  + felmásol: ☺
* **Kernel telepítése**
  + *X86:* 
    - kernel berakása a partícióra
    - bootmanager telepítése, elkészítése 🡪 az eszköz bootolható, be tudja tölteni a kernelt
  + *ARM, MIPS*
    - belső flash
      * kernel áttolása az eszköz RAMjába (vannak erre programok)
      * flash-ben megfelelő mennyiségű hely felszabadítása
      * kernel bepakol a flash-be
      * boot opciók beállítása
    - SD/MC/CF kártyán particionálás, majd rápakolás
  + *architektúra függő lépések:* gyártói dokumentáció

## Keresztfordítás (kézi)

* ***szükséges programok, és keresztfordí***tó: sysroots könyvtárban
  + keresztfordítók
  + fejlesztői könyvtárak
  + headerek
  + segédprogramok
  + konfigurációs állományok
    - egyszerűbb paramétereés: pkg-config program

## SDK készítése

* ezt telepítve a fejlesztők gépére képesek lesznek a keresztfordítások elvégzésére a teljes Yocto Project nélkül
* tartalmazza
  + keresztforító
  + szükséges fejlesztői könyvtárak
  + environment setup

## Virtuális gép generálása

* hasznos eszköz a fejlesztés során
* **virtuális gép imaget** lehet készíteni: x86-ost emuláls 🡪 a generálás várhatóan a csomagok újrafordításával jár
* **QEMU** – x86-os gép szimulálása a CPU architekturális eltérése nélkül
* **vmdk** állomány generálása – VMWare Player