

Yocto Project 시작 가이드

문서 버전: 1.0
최종 수정일: 2025-10-16
작성자: 김명환

Yocto Project 시작 가이드

목차

1. Yocto Project 개요
2. 개발 환경 구축
3. 핵심 개념 이해
4. MVP 프로젝트 실습
5. 빌드 결과물 확인

1. Yocto Project 개요

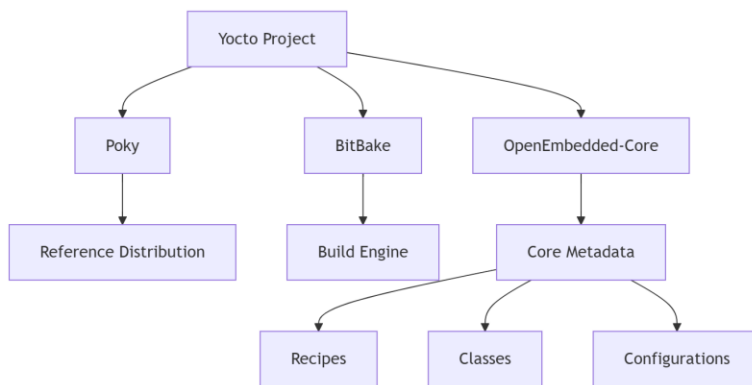
1.1. Yocto란 무엇인가

Yocto Project(옥토 프로젝트)는 임베디드 리눅스 배포판을 만들기 위한 오픈소스 협업 프로젝트입니다. 하드웨어 아키텍처에 관계없이 커스텀 리눅스 기반 시스템을 구축할 수 있는 템플릿, 도구, 메서드를 제공합니다.

1.2. 주요 특징과 장점

- 크로스 컴파일레이션(Cross-compilation) 지원
- 재현 가능한 빌드(Reproducible builds) 보장
- 모듈러 아키텍처(Modular architecture) 채택
- 광범위한 하드웨어 지원
- 라이선스 컴플라이언스(License compliance) 관리

1.3. 기본 구성 요소



2. 개발 환경 구축

2.1. 시스템 요구사항

- 최소 디스크 공간: 50GB 이상 (권장 100GB)
- RAM: 최소 4GB (권장 8GB 이상)
- CPU: 멀티코어 프로세서 권장
- OS: Ubuntu 24.04 LTS (이미 준비됨)

2.2. Ubuntu 24.04 환경 설정

```
# 시스템 업데이트
sudo apt update && sudo apt upgrade -y

# 로케일 설정 (UTF-8 필수)
sudo locale-gen en_US.UTF-8
export LANG=en_US.UTF-8
```

2.3. 필수 패키지 설치

```
# Yocto 빌드를 위한 필수 패키지 설치
sudo apt install -y \
    gawk wget git diffstat unzip texinfo gcc build-essential \
    chrpath socat cpio python3 python3-pip python3-pexpect \
    xz-utils debianutils iputils-ping python3-git python3-jinja2 \
    python3-subunit zstd liblz4-tool file locales libacl1 \
    libc6-dev-amd64-cross

# 추가 유틸리티
sudo apt install -y \
    libssl1.2-dev xterm lz4 zstd curl
```

3. 핵심 개념 이해

3.1. 레이어 (Layers)

레이어는 관련된 메타데이터의 컬렉션(collection)입니다. 각 레이어는 특정 기능이나 하드웨어 지원을 제공합니다.

```
meta/          # 코어 레이어
meta-poky/     # Poky 디스트리뷰션 레이어
meta-yocto-bsp/ # BSP 레이어
```

3.2. 레시피 (Recipes)

레시피(.bb 파일)는 소프트웨어를 빌드하는 방법을 정의합니다. 기본 구조:

```
DESCRIPTION = "패키지 설명"
LICENSE = "라이선스 종류"
SRC_URI = "소스 위치"
```

```
do_compile() {
    # 컴파일 명령
}
```

```
do_install() {
    # 설치 명령
}
```

3.3. 비트베이킹 (BitBake)

BitBake는 Yocto의 태스크 실행 엔진입니다. 파이썬으로 작성되었으며, 레시피를 파싱(parsing)하고 의존성을 해결합니다.

3.4. 메타데이터 (Metadata)

메타데이터는 빌드 시스템에 대한 정보를 포함합니다:

- Configuration files (.conf)
- Recipes (.bb, .bbappend)
- Classes (.bbclass)

4. MVP 프로젝트 실습

4.1. Poky 다운로드

```
# 작업 디렉토리 생성
mkdir ~/yocto-test && cd ~/yocto-test

# Poky 레포지토리 클론 (Scarthgap 24.0.7 LTS)
git clone -b scarthgap git://git.yoctoproject.org/poky.git
cd poky
```

4.2. 빌드 환경 초기화

```
# 빌드 환경 스크립트 실행
source oe-init-build-env build-test

# 현재 디렉토리가 build-test 로 변경됨
pwd # ~/yocto-test/poky/build-test
```

4.3. 첫 번째 이미지 빌드

4.3.1. 기본 설정 확인

```
# local.conf 파일 확인
cat conf/local.conf | grep "^MACHINE"
# 기본값: MACHINE ?= "qemux86-64"

# 병렬 빌드 설정 (CPU 코어 수에 따라 조정)
echo 'BB_NUMBER_THREADS = "4"' >> conf/local.conf
echo 'PARALLEL_MAKE = "-j 4"' >> conf/local.conf
```

4.3.2. 미니멀 이미지 빌드

```
# core-image-minimal 빌드 (가장 작은 이미지)
bitbake core-image-minimal

# 첫 빌드는 1-3 시간 소요 (인터넷 속도와 CPU 성능에 따라 다름)
```

4.4. QEMU 로 테스트

```
# QEMU 에뮬레이터로 빌드된 이미지 실행
runqemu qemux86-64 core-image-minimal

# QEMU 내에서:
# root 로 로그인 (비밀번호 없음)
# 기본 명령어 테스트
uname -a
cat /etc/os-release
ps aux

# QEMU 종료: Ctrl-A 후 X
```

5. 빌드 결과물 확인

5.1. 생성된 이미지 분석

```
# 빌드 결과물 위치
cd tmp/deploy/images/qemux86-64/

# 생성된 파일 확인
ls -lh core-image-minimal*

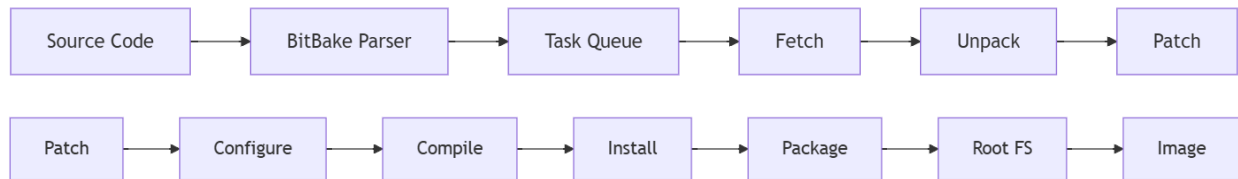
# 주요 파일:
# - core-image-minimal-qemux86-64.rootfs.ext4 # 루트 파일시스템
# - core-image-minimal-qemux86-64.qemuboot.conf # QEMU 설정
# - bzImage-qemux86-64.bin # 커널 이미지
```

5.2. 패키지 구조 이해

```
# 설치된 패키지 목록 확인
cd ../../rpm/
ls -la

# 매니페스트 파일 확인
cd ../images/qemux86-64/
cat core-image-minimal-qemux86-64.manifest | head -20
```

빌드 프로세스 다이어그램



트러블슈팅 팁

일반적인 문제 해결

1. 디스크 공간 부족

```
# 빌드 캐시 정리
bitbake -c cleansstate core-image-minimal
```

2. 네트워크 문제

```
# 프록시 설정 (필요시)
export http_proxy=http://proxy.example.com:8080
export https_proxy=http://proxy.example.com:8080
```

3. 빌드 로그 확인

```
# 상세 로그 위치
ls tmp/work/*/core-image-minimal/*/temp/log.*
```

다음 단계

1. 커스텀 레이어 생성

```
bitbake-layers create-layer meta-custom
```

2. 자체 레시피 작성

간단한 “Hello World” 애플리케이션 추가

3. 이미지 커스터마이제이션

추가 패키지 포함

설정 파일 수정

용어 목록

용어	설명
BitBake	Yocto의 태스크 실행 엔진, 메타데이터를 파싱하고 빌드 태스크를 실행
BSP (Board Support Package)	특정 하드웨어 보드를 지원하기 위한 드라이버와 설정
Cross-compilation	타겟 아키텍처와 다른 호스트에서 컴파일하는 과정
Image	타겟 디바이스에 배포될 완전한 루트 파일시스템
Layer	관련된 레시피, 설정, 클래스를 포함하는 메타데이터 컬렉션
Manifest	이미지에 포함된 모든 패키지의 목록
Metadata	빌드 시스템을 제어하는 레시피, 설정, 클래스 파일들
OE-Core (OpenEmbedded-Core)	Yocto와 OpenEmbedded가 공유하는 핵심 메타데이터
Poky	Yocto Project의 레퍼런스 디스트리뷰션

QEMU	하드웨어 에뮬레이터, 빌드된 이미지를 테스트하는 데 사용
Recipe	소프트웨어 패키지를 빌드하는 방법을 정의하는 파일 (.bb)
Rootfs	루트 파일시스템, 리눅스 시스템의 기본 디렉토리 구조
Sstate-cache	빌드 속도 향상을 위한 공유 상태 캐시
Task	레시피 내에서 실행되는 개별 빌드 단계 (fetch, compile 등)
Toolchain	크로스 컴파일을 위한 컴파일러와 도구 모음