

# Embedded Linux on Zynq using Vivado Lab3

Innova Lee(이상훈)  
gcccompil3r@gmail.com

## Introduction

PetaLinux 도구를 사용하면 응용 프로그램을 쉽게 작성하고 임베디드 Linux 이미지에 빌드 할 수 있다.

대부분의 경우 임베디드 Linux 가 실행되는 임베디드 Target 시스템이 아닌

일반(Host) 컴퓨터 시스템에서 응용 프로그램을 작성한다.

이 경우 크로스 컴파일 이 필요하다.

PetaLinux 도구를 사용하면 데스크탑 Linux 에서 임베디드 Linux 응용 프로그램을 크로스 컴파일 할 수 있다.

PetaLinux 툴은 TCF(Target Communication Framework) Agent 를 사용하는 System Debugger 로

Zynq All Programmable SoC 사용자 Application 디버깅을 지원한다.

TCF 를 사용하면 Target 에서 원격으로 실행중인 App 을 디버깅 할 수 있다.

이전 Lab 에서는 ARM Cortex-A9 프로세서 System 전용 표준 임베디드 Linux Target 을 빌드하는 방법을 배웠다.

임베디드 리눅스 배포판에는 많은 유용한 응용 프로그램과 유틸리티가 포함되어 있지만

시스템 요구 사항을 충족 시키려면 자체 응용 프로그램을 작성해야 할 가능성이 높다.

이 Lab 의 목표는 ARM Cortex-A9 MPcore 임베디드 리눅스에서 자신 만의 응용 프로그램을 만들고, 개발하고, 빌드하고, 실행하고 디버깅 할 수 있도록 돕는 것이다.

예제 응용 프로그램은 간단하다.

그러나 개념과 원칙은 모두 크고 복잡한 응용 프로그램에 직접 적용된다.

이 랩은 이전 랩에서 배운 기술, 특히 Linux 시스템 구축 및 부팅,

ARM Cortex-A9 MPcore Linux 시스템 로그인에 직접 적용된다.

이 과정에 대해 의문이 존재할 경우 이전 Lab 을 참조하라.

## Objectives

이 Lab 을 완료하면 아래를 수행 할 수 있다:

- PetaLinux 도구로 간단한 사용자 응용 프로그램 만들기
- 임베디드 리눅스로 크로스 컴파일하여 새로운 사용자 애플리케이션을 구축한다.
- 임베디드 리눅스에서 응용 프로그램 실행하기
- 시스템 디버거를 사용하여 응용 프로그램 디버깅하기

## Preparation

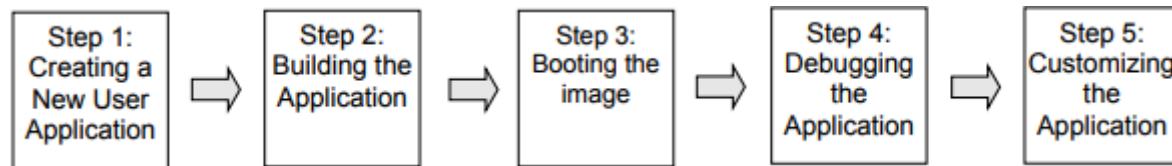
이것이 첫 번째 Lab 인 경우 환경 설정 방법에 대한 필수 준비 정보는 Lab 1 의 "Before You Start" 섹션을 참조하라.

워크 스테이션이 다시 시작되거나 로그 아웃되면 아래 명령을 실행하여 Host 에서 DHCP 서버를 시작한다:

```
[host]$ sudo service isc-dhcp-server restart
```

자세한 내용은 Lab 1 의 "Initializing the Workshop Environment" 섹션을 참조하라.

## General Flow for this Lab



## Creating a New User Application

ARM Cortex-A9 MPcore 시스템에서 사용자 응용 프로그램을 실행하려면 크로스 컴파일하고 Host 시스템의 임베디드 리눅스 이미지에 빌드해야 한다. PetaLinux 도구를 사용하면 이러한 단계를 쉽게 수행 할 수 있다.

1-1. 경로를 프로젝트 디렉토리로 변경한다.

1-1-1. 아래 명령을 실행하여 프로젝트 디렉토리 경로를 만들고 변경한다.

```
[host]$ mkdir ~/emblnx/labs/lab3
```

```
[host]$ cd ~/emblnx/labs/lab3
```

1-2. petalinux-create 명령을 사용하여 새로운 임베디드 리눅스 플랫폼을 만들고 플랫폼을 선택한다.

1-2-1. 아래 명령을 실행하여 새로운 petalinux 프로젝트를 만든다:

```
[host]$ petalinux-create -t project -s /opt/pkg/ZYBO_petalinux_v2015_4.bsp
```

이 명령은 ~/emblnx/labs/lab3 아래에 ZYBO\_petalinux\_v2015\_4 라는 PetaLinux 소프트웨어 프로젝트 디렉토리를 만든다.

1-2-2. 디렉토리를 PetaLinux 프로젝트로 변경한다:

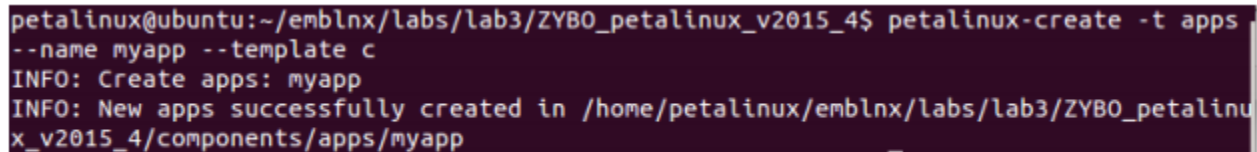
```
~/emblnx/labs/lab3/ZYBO_petalinux_v2015_4
```

1-3. 새로운 응용 프로그램을 만든다.

PetaLinux 도구를 사용하면 C 또는 C++ 전용 사용자 응용 프로그램 템플릿을 만들 수 있다. 이러한 템플릿에는 응용 프로그램 소스 코드 및 메이크 파일이 포함되어 있으므로 Target 에 맞게 응용 프로그램을 쉽게 구성하고 컴파일하여 루트 파일 시스템에 설치할 수 있다.

1-3-1. 아래 명령을 입력하여 PetaLinux 프로젝트 내에 새로운 사용자 응용 프로그램을 생성한다:

```
[host]$ petalinux-create -t apps --name myapp --template c
```



```
petalinux@ubuntu:~/emblnx/labs/lab3/ZYBO_petalinux_v2015_4$ petalinux-create -t apps  
--name myapp --template c  
INFO: Create apps: myapp  
INFO: New apps successfully created in /home/petalinux/emblnx/labs/lab3/ZYBO_petalinu  
x_v2015_4/components/apps/myapp
```

Figure 1. Creating a new application

새로 만든 응용 프로그램은 <project-root>/components/apps/myapp 디렉토리에서 찾을 수 있다. 여기서 <project-root> 는 ~/emblnx/labs/lab3/ZYBO\_petalinux\_v2015\_4 에 해당한다.

C++ 응용 프로그램 템플릿을 만들려면 --template c++ 옵션을 사용한다.

1-3-2. 새로 생성 된 응용 프로그램 디렉토리로 변경한다.

```
[host]$ cd <project-root>/components/apps/myapp
```

아래와 같은 PetaLinux 템플릿 생성 파일이 표시된다:

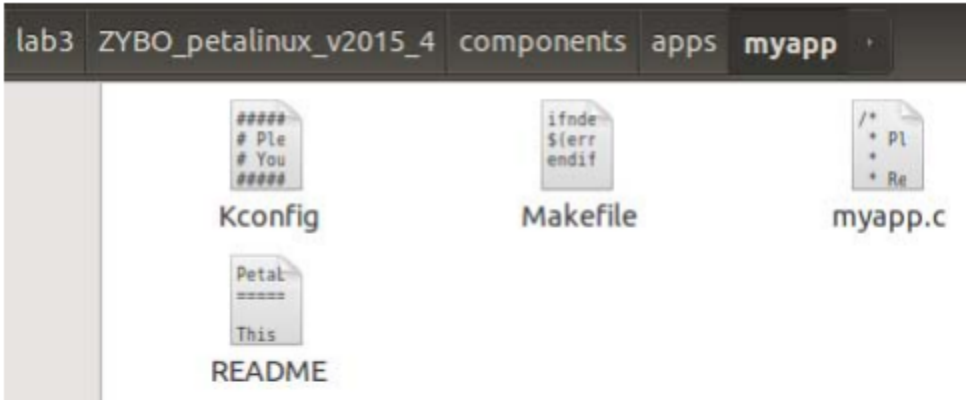


Figure 2. Files Generated After Creating a New Application

Template	Description
Kconfig	Configuration file template. This file controls how your application is integrated into the PetaLinux menu configuration system and allows you to add configuration options for your own application to control how it is built or installed.
Makefile	Compilation file template. This is a basic makefile containing targets to build and install your application into the root file system. This file needs to be modified when you add additional source code files to your project.
README	A file to introduce how to build the user application.
myapp.c for C or myapp.cpp for C++	Simple application program in either C or C++, depending upon your choice. These files will be edited or replaced with the real source code for your application.

## Building the Application into the Embedded Linux

새로운 응용 프로그램을 만들었으면 다음 단계는 컴파일하고 빌드하는 것이다.

2-1. 빌드 프로세스에 포함될 새 애플리케이션을 선택하라.

응용 프로그램은 기본적으로 활성화되어 있지 않다.

2-1-1. 현재 작업 디렉토리(프로젝트 디렉토리)가 ~/emblnx/labs/lab3/ZYBO\_petalinux\_v2015\_4 에 해당한다.

2-1-2. 아래 명령을 입력하여 rootfs 구성 메뉴를 실행한다.

```
[host]$ petalinux-config -c rootfs
```

참고: 터미널 창이 최소 80 열 이상인지 확인하라.

linux/rootfs 설정 메뉴가 열린다.

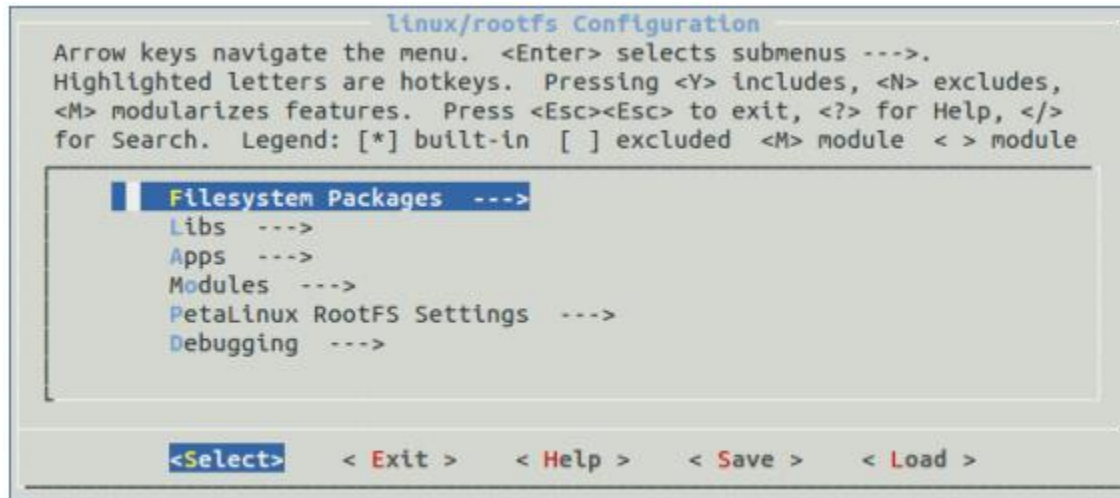


Figure 3. The linux/rootfs Configuration menu

2-1-3. 아래쪽 화살표 키를 눌러 메뉴를 Apps 로 스크롤 한다.

2-1-4. <Enter> 키를 눌러 Apps 하위 메뉴로 이동한다.

새로운 응용 프로그램 myapp 이 메뉴에 나열된다.

2-1-5. myapp 으로 이동하고 <Y> 키를 눌러 응용 프로그램을 선택하라.

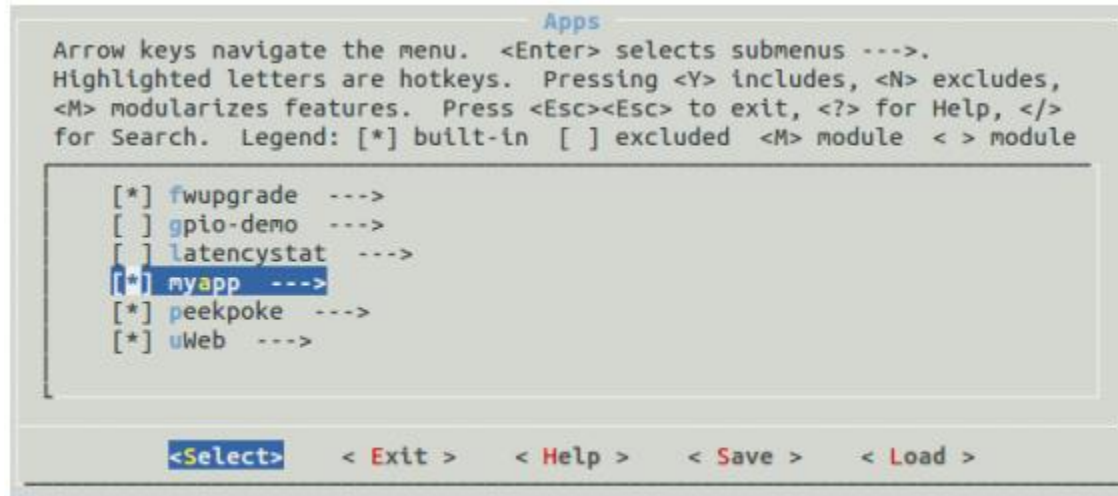


Figure 4. Selecting the new application myapp

2-1-6. <Enter> 키를 눌러 myapp 옵션을 연다.

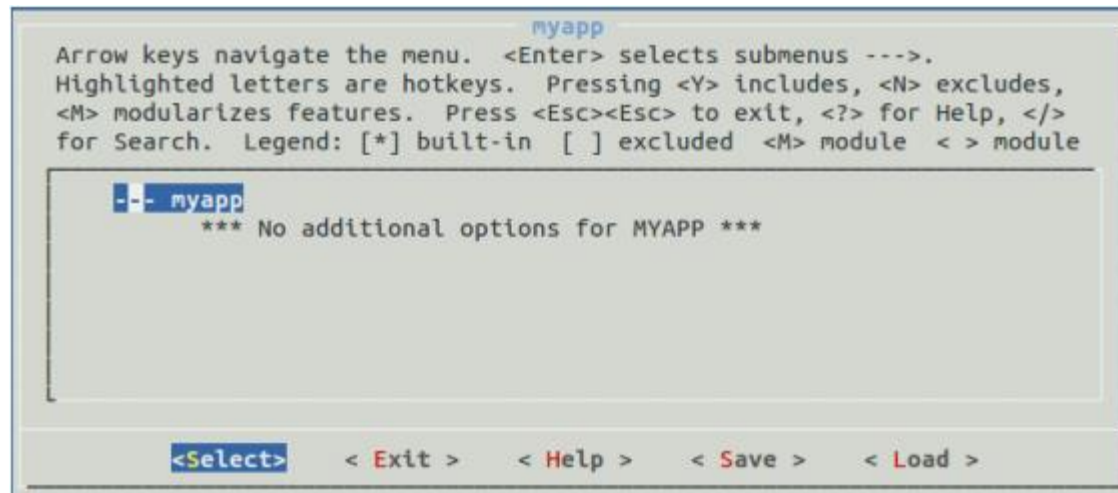


Figure 5. No additional options for myapp sub-menu

기본적으로 myapp 에 대한 추가 옵션은 없다.

고급 사용자는 myapp 디렉토리의 Kconfig 파일을 수정하여 사용자 정의 옵션을 추가 할 수 있다.

2-1-7. 메인 메뉴로 돌아가기 위해 Exit 를 선택하고 누른다.

2-2. 디버깅 할 수 있는 응용 프로그램을 빌드하라.

2-2-1. linux/rootfs 설정 메뉴를 아래로 스크롤하여 디버깅 한다.

2-2-2. 디버깅 하위 메뉴를 선택하고 디버그 가능한 응용 프로그램 빌드가 선택되었는지 확인하라.  
디버그 가능한 응용 프로그램 빌드를 선택하려면 Y 를 클릭한다.

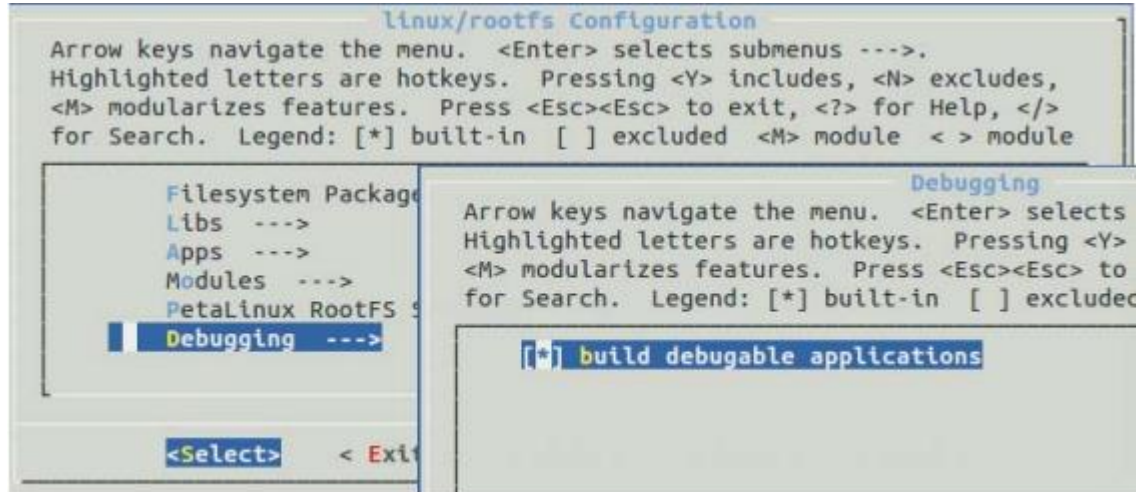


Figure 6. Selecting build debug-able applications

2-2-3. 메인 메뉴로 돌아가기 위해 Exit 를 선택하고 누른다.

2-3. 디버깅 지원을 위해 TCF Agent 를 활성화 한다.

PetaLinux 툴은 TCF Agent 로 Zynq All Programmable SoC 사용자 애플리케이션을 디버깅하는 것을 지원한다.

2-3-1. 파일 시스템 패키지를 선택하라.

2-3-2. 파일 시스템 패키지 메뉴에서 base 를 선택한다.

2-3-3. base 메뉴에서 아래로 스크롤 하여 tcf-agent 를 선택한다.



2-3-4. Y 를 클릭하여 tcf-agent 를 활성화 한다.

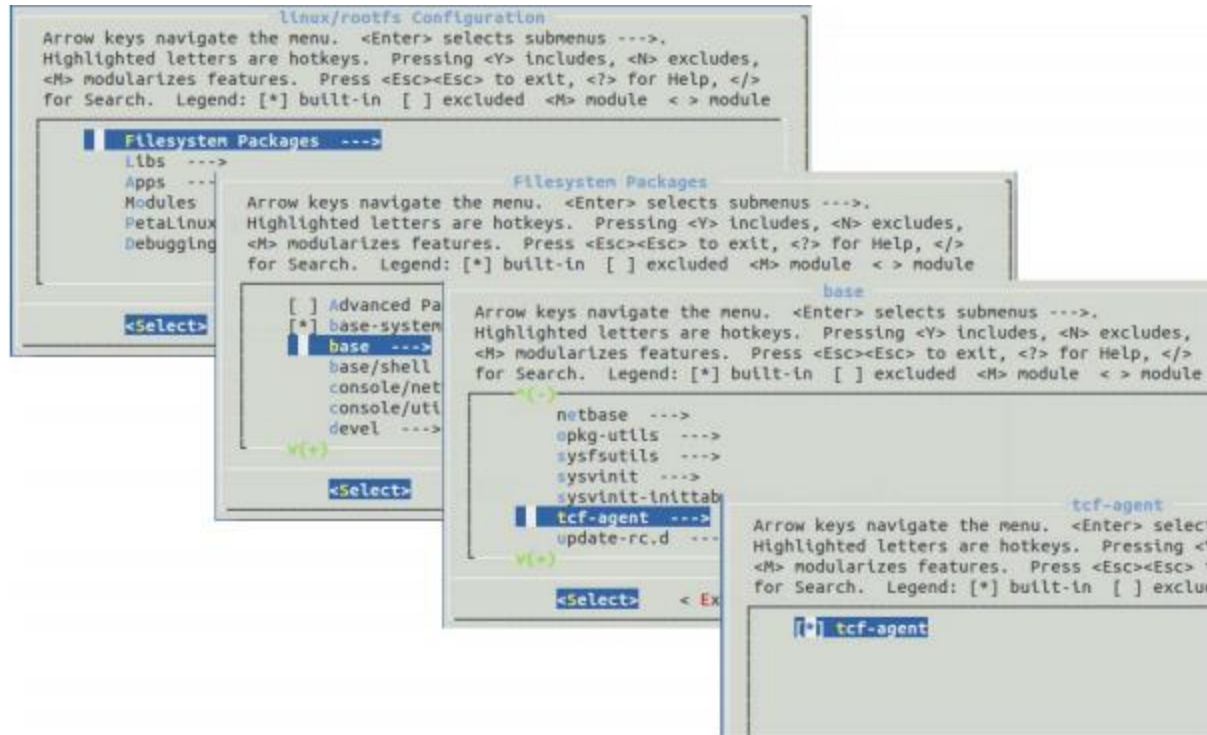


Figure 7. Enabling the TCF Agent

2-3-5. 메인 메뉴로 돌아가기 위해 Exit 를 선택하고 누른다.

2-3-6. linux/rootfs 설정 메뉴를 종료한다.

2-3-7. 새 구성을 저장하려면 <Yes> 를 선택한다.  
구성 변경 사항이 적용되는데 몇 초가 걸린다.  
명령 콘솔에서 쉘 프롬프트로 돌아갈 때까지 기다린다.

## 2-4. Image 를 Build 한다.

### 2-4-1. 아래 명령을 입력하여 Image 를 빌드한다:

```
[host]$ petalinux-build
```

QEMU 시뮬레이터를 사용하면 HW 를 전혀 사용하지 않고도 SW 응용 프로그램을 개발하고 디버깅 할 수 있다.

### 3-1. QEMU 에서 응용 프로그램을 실행한다.

#### 3-1-1. QEMU 를 통해 새로 빌드 된 PetaLinux 이미지를 부팅하려면 아래 명령을 입력한다.

```
#petalinux-boot --qemu --kernel
```

#### 3-1-2. 시스템이 부팅 된 후 로그인 이름과 암호로 root 를 입력하여 시스템에 로그인 한다.

#### 3-1-3. QEMU 콘솔의 /bin 디렉토리를 검사한다.

```
#ls /bin | grep myapp
```

myapp 응용 프로그램이 있는지 확인해야 한다.

#### 3-1-4. QEMU 콘솔에서 myapp 응용 프로그램을 실행한다.

```
# myapp 1
The following is the output of the command:
Hello, PetaLinux World!
cmdline args:
myapp
1
```

#### 3-1-5. Ctrl + a 키를 누른 다음 x 키를 눌러 QEMU 를 종료한다.

# Debugging the Application Using System Debugger in Board

4-1. XSDK 를 시작하고 작업 공간을 만든다.

4-1-1. 새 터미널을 연다.

4-1-2. lab3 아래에 작업 디렉토리를 작성하려면 아래 명령을 입력한다.

```
[host]$ mkdir ~/emblnx/labs/lab3/workspace
```

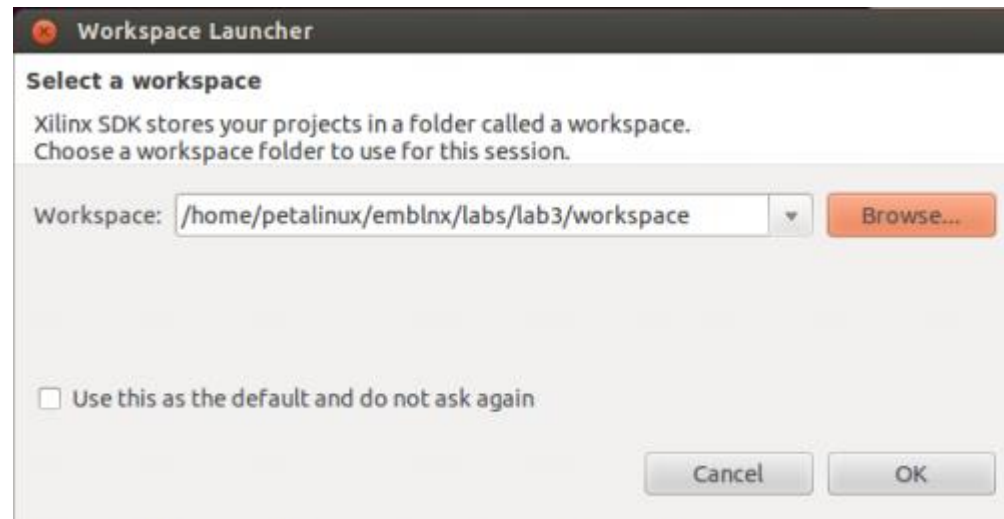
4-1-3. xsdk 를 시작하려면 아래 명령을 입력한다.

```
[host]$ xsdk
```

프로그램이 시작되지 않으면 아래 명령을 실행하여 설정을 가져온다.

```
[host]$ source /opt/pkg/Xilinx/Vivado/2015.4/settings64.sh
```

4-1-4. ~/emblnx/labs/lab3/workspace 를 작업 공간 디렉토리로 선택한다.



4-1-5. OK 를 클릭한다.

Figure 8. Setting up the Workspace Environment path

4-1-6. Welcom 화면이 열려 있으면 닫는다.

4-2. 하드웨어 플랫폼 명세를 만든다.

4-2-1. File -> New -> Project 를 선택한다.

4-2-2. 팝업 창에서 Xilinx -> Hardware Platform Specification 을 선택한다.

4-2-3. Next 를 클릭한다.

4-2-4. 프로젝트 이름으로 zynq\_hw\_platform 을 입력한다.

4-2-5. Target Hardware Specification 영역에서 ~/emblnx/sources/lab3/ 디렉토리로 이동하여 zybo\_wrapper.hdf 를 선택한다.

4-2-6. OK 를 클릭한다.

4-2-7. Finish 를 클릭한다.

4-3. SD 카드에 있는 BOOT.BIN 파일이 미리 빌드 된 디렉토리에 복사되었는지 확인한다.

4-3-1. 사전 빌드 된 BOOT.BIN 파일이 SD 카드에 있는지 확인한다.

Lab 2 를 마지막 Lab 으로 사용했다면 SD 카드를 변경할 필요가 없다.

그렇지 않다면 BOOT.BIN 을 ~/emblnx/lab3/ZYBO\_petalinux\_v2015\_4/prebuilt/linux/images 디렉토리에서 SD 카드로 복사한다.

4-3-2. SD 카드를 Target Board 에 다시 넣는다.

4-4. Host 에서 DHCP 서버를 실행한다.

4-4-1. DHCP 서버를 실행한다.

```
[host]$ sudo service isc-dhcp-server restart
```

4-5. 보드의 전원을 켜고 Serial 포트 터미널을 설정하라.

4-5-1. 보드의 전원을 켜다.

4-5-2. /dev/ttyUSB1 이 read/write 접근으로 설정되어 있는지 확인한다.

```
[host]$ sudo chmod 666 /dev/ttyUSB1
```

4-5-3. dashboard 의 검색 필드에 Serial 포트를 입력한다.

4-5-4. Serial 포트 터미널 응용 프로그램을 선택한다.

Board 를 리셋(BTN7)하여 부팅 정보를 다시 한 번 볼 수 있다.

4-6. 새로운 리눅스 이미지를 보드에서 부팅한다.

4-6-1. GtkTerm 창에서 부팅 프로세스를 본다.

4-6-2. GtkTerm 창에 다음과 유사한 메시지가 나타나면 아무 키나 눌러 자동 부팅을 중지한다.

```
U-Boot 2015.07 (Jan 21 2016 - 07:27:49 +0000)

DRAM: ECC disabled 512 MiB
MMC: zynq_sdhci: 0
SF: Detected S25FL128S_64K with page size 256 Bytes, erase size 64 KiB, total 16 MiB
*** Warning - bad CRC, using default environment

In: serial
Out: serial
Err: serial
Net: Gem.e000b000
U-BOOT for ZYBO_petalinux_v2015_4

Hit any key to stop autoboot: 0
U-Boot-PetaLinux> █
```

**Figure 9. Stopping the Autoboot**

4-6-3. uboot 부팅 중 "DHCP client bound to address" 메시지가 표시되지 않으면 dhcp 를 실행하여 IP 주소를 얻어야 한다.

```
U-Boot-PetaLinux> dhcp
```

4-6-4. u-boot 콘솔에서 아래 명령을 실행하여 TFTP 서버 IP 를 Host IP 로 설정한다.

```
U-Boot-PetaLinux> set serverip 192.168.1.1
```

4-6-5. u-boot 콘솔에서 아래 명령을 실행하여 TFTP 를 사용하여 새 이미지를 다운로드하고 부팅한다.

```
U-Boot-PetaLinux> run netboot
```

이 명령은 Host 의 /tftpboot 에서 ARM Cortex-A9 MPcore 시스템의 주 메모리로 image.ub 파일을 다운로드하고 이미지로 시스템을 부팅한다.

4-6-6. 시스템이 부팅 된 이후 로그인 이름과 암호로 root 를 입력하여 시스템에 로그인한다.

4-6-7. TCF Agent 가 실행 중인지 확인한다.

4-6-8. GtkTerm 창에 아래 명령을 입력하여 보드의 IP 주소를 확인한다.

```
#ifconfig
```

4-6-9. eth0 IP 주소를 확인한다.

4-7. 새로운 Debug 구성을 만들고 Target 설정을 구성한다.

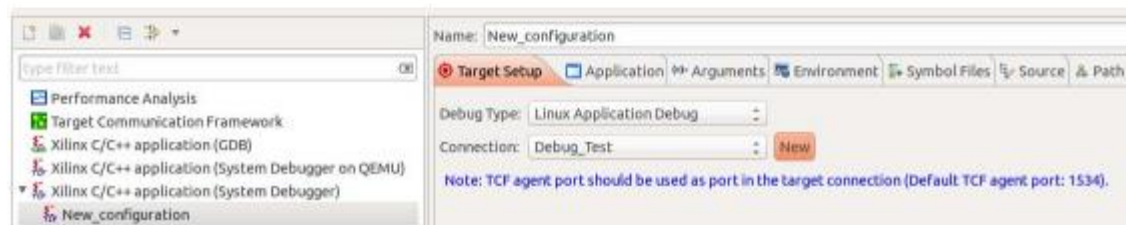
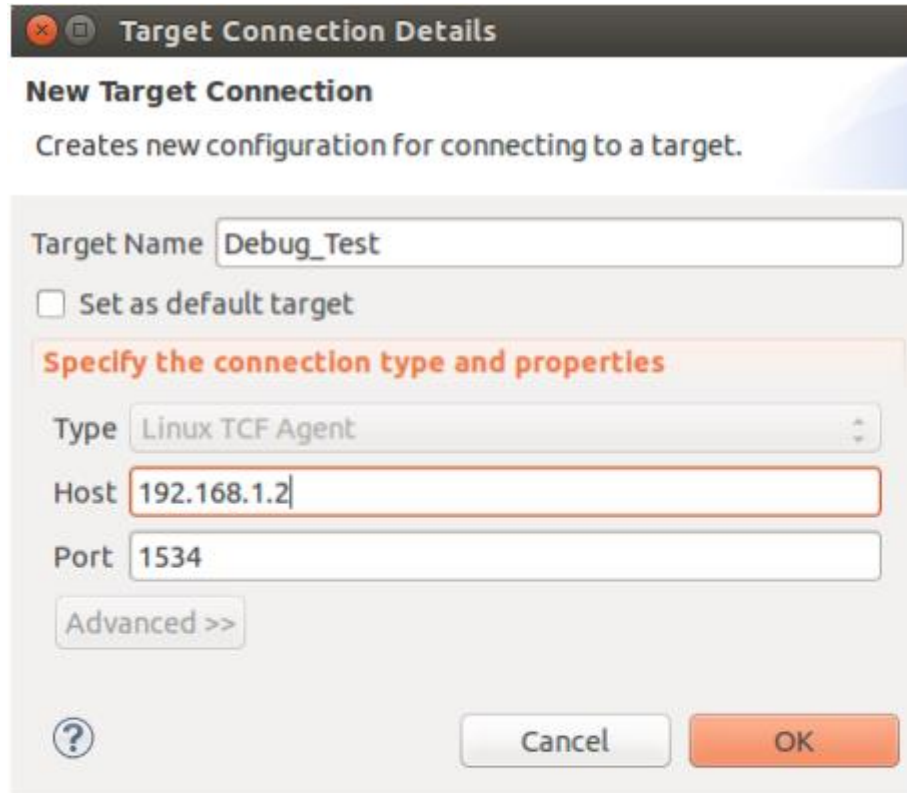
4-7-1. SDK 에서 Run -> Debug Configurations ... 를 선택한다.

4-7-2. Xilinx C/C++ Application(System Debugger)를 더블 클릭하여 새 구성을 만든다.

4-7-3. Debug 유형으로 Linux Application Debug 를 선택한다.

4-7-4. Connection 의 New button 을 클릭한다.

4-7-5. Target 이름 필드에 Debug\_Test 를 입력하고 Host 필드에 192.168.1.1 을 입력 한 다음 OK 를 클릭한다.



4-8. 로컬 및 원격 파일 경로를 모두 구성한다.

4-8-1. Application 탭을 선택한다.

4-8-2. Browse 를 클릭하여 프로젝트 디렉토리에서 컴파일 된 응용 프로그램에 로컬 파일 경로를 설정하고 아래 폴더를 찾는다:  
<project-root>/build/linux/rootfs/apps/myapp/myapp

4-8-3. 원격 파일 경로를 응용 프로그램을 찾을 수 있는 Target 파일 시스템의 위치로 설정한다:  
/bin/myapp

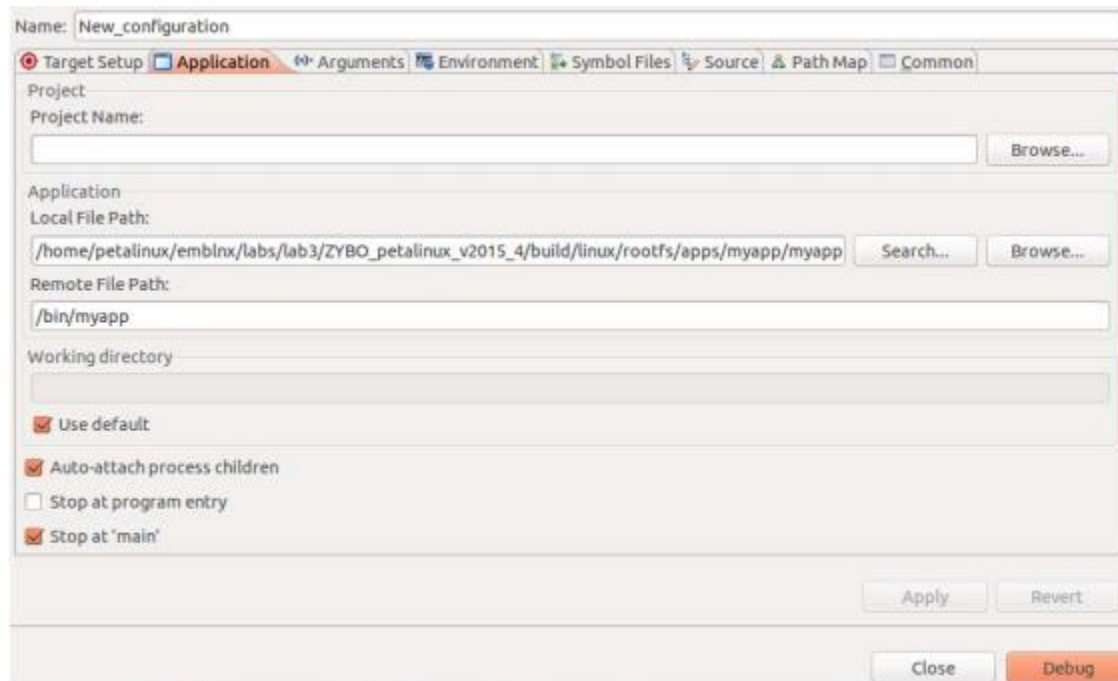


Figure 11. Configuring the Application File paths

4-8-4. Apply 를 클릭한다.



## 4-9. 프로그램을 디버그한다.

### 4-9-1. Debug 를 클릭한다.

### 4-9-2. Perspective 스위치를 확인하려면 Yes 를 클릭한다. Debug Perspective 가 열린다.

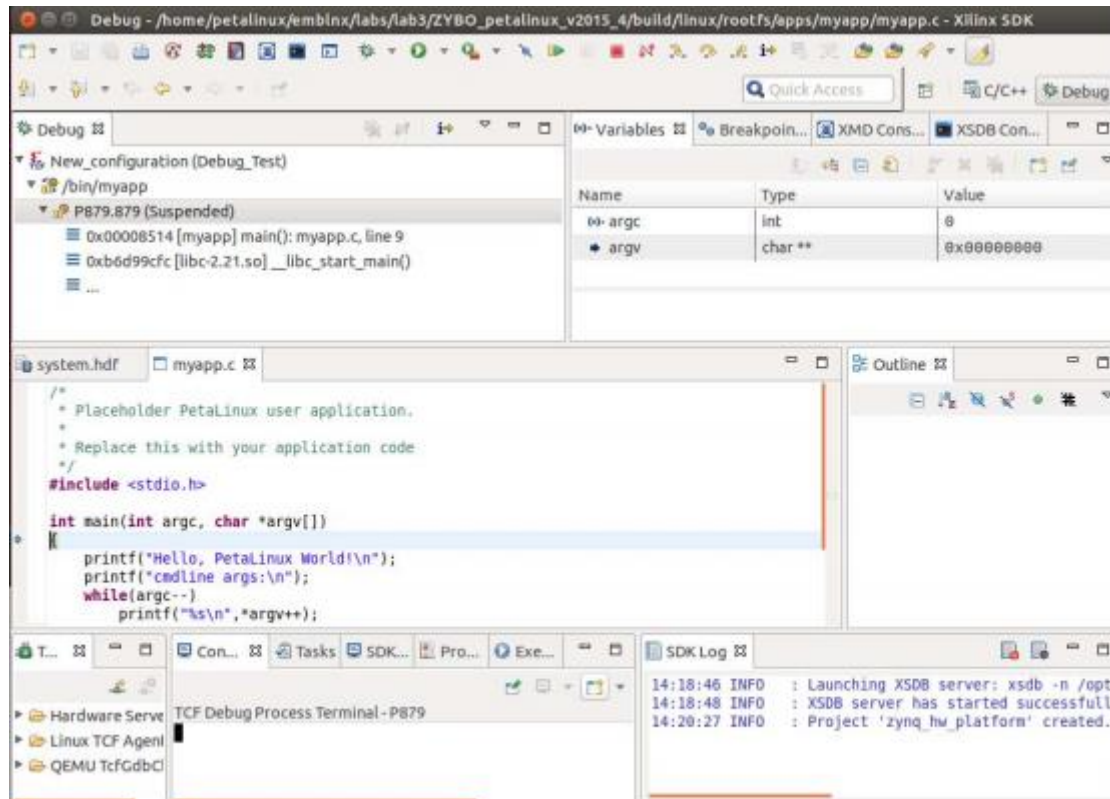


Figure 12. Debug perspective window

프로그램 작업은 main{} (실행되지 않음) 의 첫 번째 실행 문에서 일시 중지된다.  
현재 함수의 지역 변수는 변수 탭에 표시된다.

4-9-3. Window -> Show View -> Disassembly 를 선택한다.

4-9-4. 줄 번호 11 을 두 번 클릭하여 breakpoint 를 설정한다(확인 표시가 나타난다)

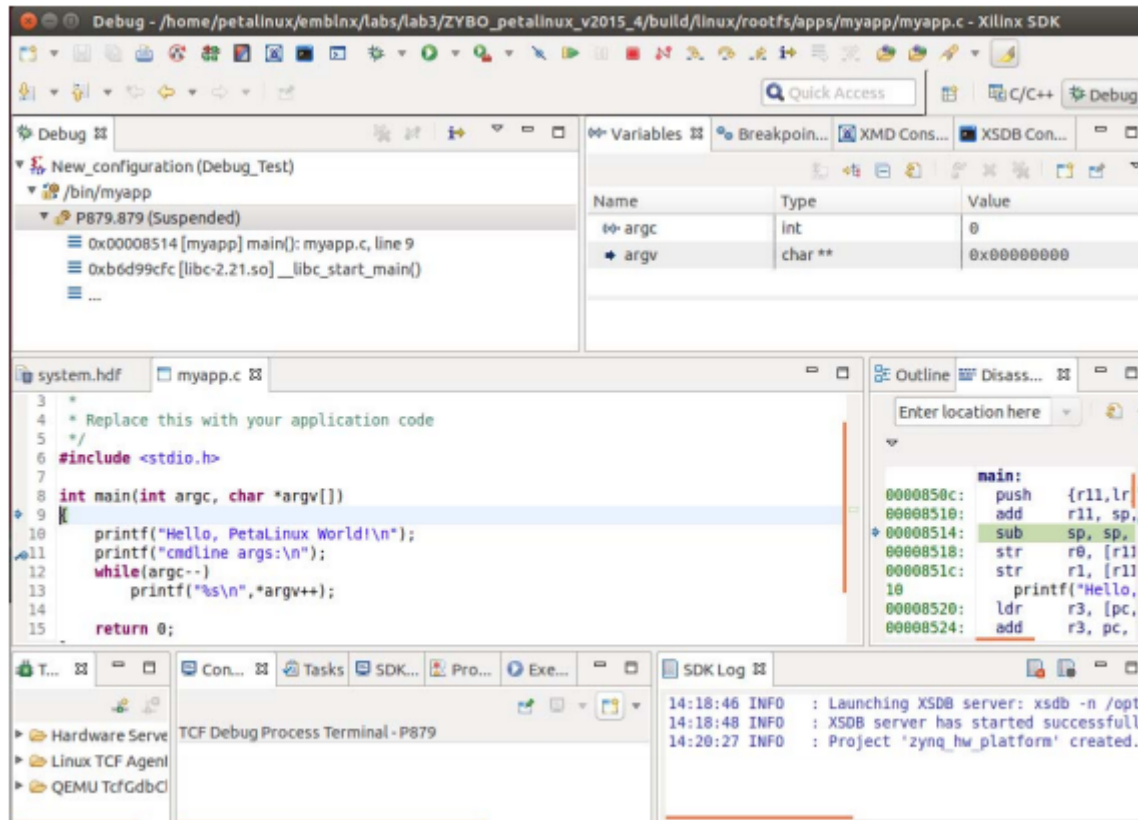


Figure 13. Breakpoint set at line number 11

참고: 행 번호가 표시되지 않으면 편집기의 가장 왼쪽 열을 마우스 우클릭하고 'Show Line Numbers' 를 선택한다.

## 4-10. 프로그램을 재개한다.

### 4-10.1 Play/Resume 버튼(초록색 삼각형)을 클릭하여 프로그램을 실행한다. 프로그램은 breakpoint 까지 실행된다.

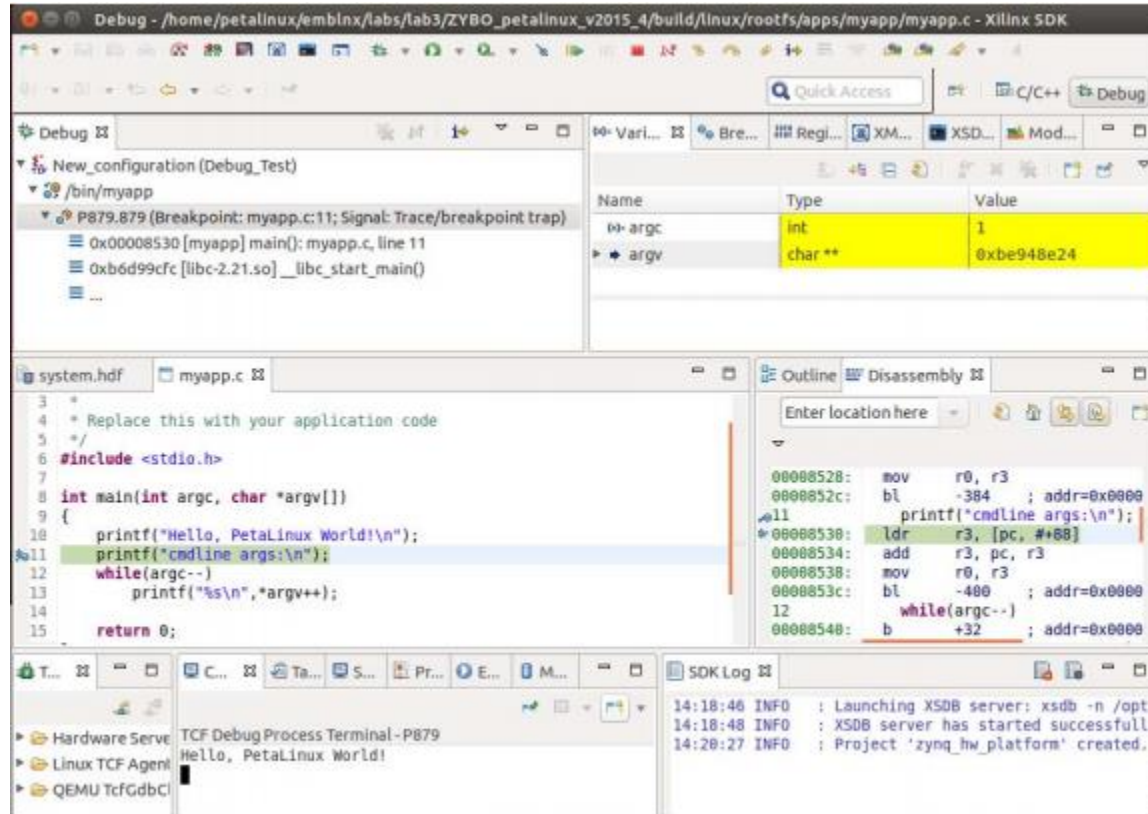


Figure 14. Program Stops at breakpoint

콘솔 창에 메시지가 표시된다.

Disconnect 버튼을 클릭하여 다른 옵션을 탐색하고 프로그램의 연결을 끊을 수 있다.

### 4-10-2. File -> Exit 를 선택하여 XSDK 도구를 닫는다.

# Customizing the Application Template

5-1. 구성 가능한 welcome 문자열을 출력하려면 Source File 을 편집하라.

5-1-1. Host 터미널에서 디렉토리를 아래와 같이 변경한다:

```
[host]$ cd ~/emblnx/labs/lab3/ZYBO_petalinux_v2015_4/components/apps/myapp
```

5-1-2. myapp.c 파일을 편집하려면 아래 명령을 입력한다:

```
[host]$ gedit myapp.c
```

5-1-3. 아래 그림에 설명된 라인(코드)를 추가한다.

```
#include <stdio.h>

int main(int argc, char *argv[])
{
    char *welcome;
    #ifdef WELCOME
        welcome=WELCOME;
    #else
        welcome="PetaLinux World!";
    #endif
    printf("Hello, %s\n",welcome );
    printf("cmdline args:\n");
    while(argc--)
        printf("%s\n",*argv++);

    return 0;
}
```

Figure 15. Customizing the myapp file

참고: welcome 의 해당 값을 사용하려면 첫 번째 printf 문을 수정해야 한다.

5-1-4. 파일을 저장하고 닫는다.

5-2. 구성 옵션을 추가하려면 사용자 응용 프로그램의 Kconfig 파일을 편집한다.

5-2-1. Kconfig 파일을 편집하려면 아래 명령을 입력한다.

```
[host]$ gedit Kconfig
```

5-2-2. 아래 그림에 설명된 코드를 추가한다.

```
if ROOTFS_COMPONENT_APPS_NAME_MYAPP
    comment "No additional options for MYAPP"

    config APPS_MYAPP_WELCOME
    string "Welcome String"
    help
    Welcome string for myapp

#    config APPS_MYAPP_OPTION0
#    bool "option0"
#    help
#        Help text
endif
```

Figure 16. Adding a Configuration Option in the Kconfig file

5-2-3. 파일을 저장하고 닫는다.

5-3. 구성 가능한 옵션을 사용자 응용 프로그램 실행 파일에 전달하도록 사용자 응용 프로그램 makefile 을 편집하라.

5-3-1. makefile 을 편집하려면 아래 명령을 입력하라.

```
[host]$ gedit Makefile
```

5-3-2. 아래 그림에 설명된 라인(코드)를 추가하라.



```
include apps.common.mk  
include $(ROOTFS_CONFIG)  
ifneq ($(CONFIG_APPS_MYAPP_WELCOME),)  
CFLAGS += -DWELCOME=\"$(CONFIG_APPS_MYAPP_WELCOME)\"  
endif  
APP = myapp
```

Figure 17. Modifying the Makefile

5-3-3. 파일을 저장하고 닫는다.

5-4. 응용 프로그램 구성 메뉴를 다시 실행한다.

5-4-1. <project-root> 디렉토리로 변경한다.

즉 ~/emblnx/labs/lab3/ZYBO\_petalinux\_v2015\_4 로 변경한다.

5-4-2. 아래 명령을 입력하라:

```
[host]$ petalinux-config -c rootfs
```

5-4-3. Apps -> myapp 을 선택한다.

myapp 하위 메뉴에서 새로운 Welcome String 구성을 볼 수 있다.

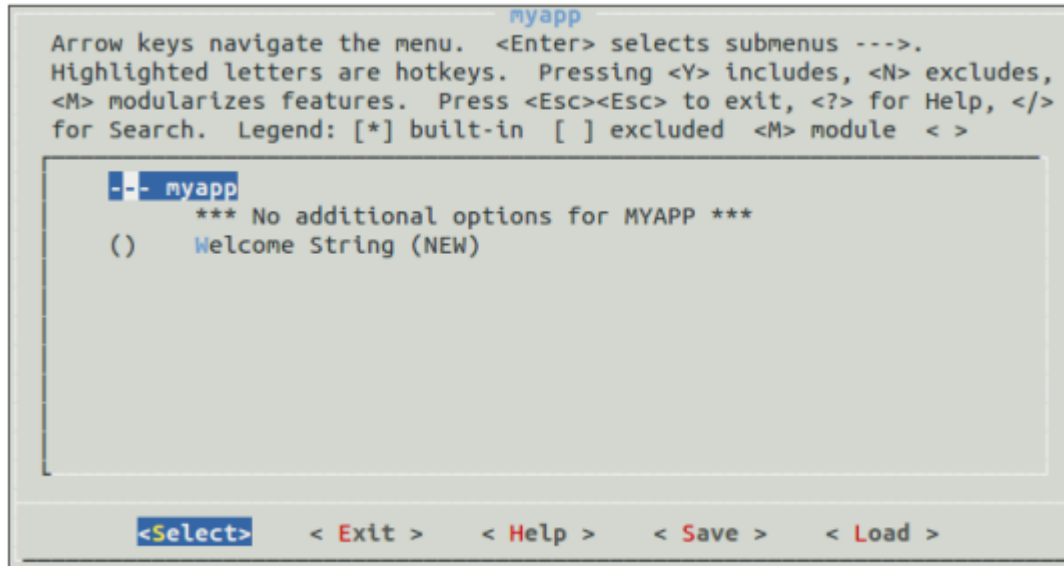


Figure 18. Welcome string configuration

5-4-4. Welcome String 옵션을 선택한다.

5-4-5. It's a user application test! 를 입력한다.

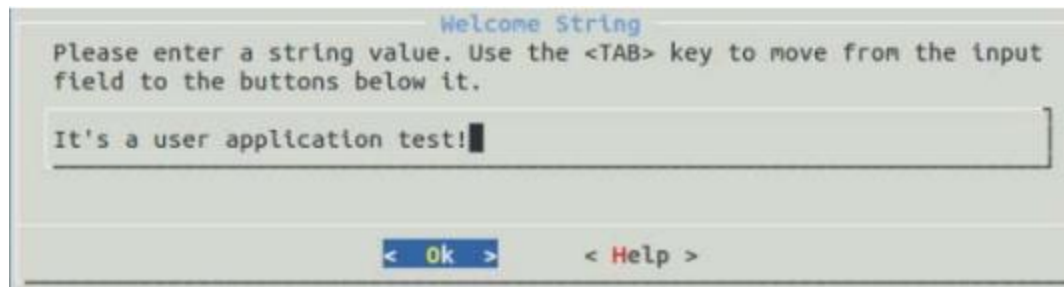


Figure 19. Entering the Welcome string

구성 변경 사항을 저장하고 종료한다.

5-5. 이미지를 다시 작성한다.

5-5-1. <project-root> 디렉토리로 변경한다.

즉 ~/emblnx/labs/lab3/ZYBO\_petalinux\_v2015\_4 로 변경한다.

5-5-2. 응용 프로그램을 다시 작성하고 시스템 이미지를 Target 으로 지정한다.

```
[host]$ petalinux-build -c rootfs/myapp -x clean
```

```
[host]$ petalinux-build -c rootfs/myapp
```

```
[host]$ petalinux-build -x package
```

5-6. QEMU 에서 응용 프로그램을 실행한다.

5-6-1. QEMU 를 통해 새로 Build 된 PetaLinux 이미지를 부팅하려면 아래 명령을 입력하라.

```
[host]$ petalinux-boot --qemu --kernel
```

5-6-2. 시스템이 부팅 된 후 시스템에 로그인한다.

5-6-3. QEMU 콘솔에서 myapp 응용 프로그램을 실행한다:

```
# myapp
The following is the output of the command:
Hello, It's a user application test!
cmdline args:
myapp
```

5-6-4. Ctrl + a 키를 누른 다음 x 키를 눌러 QEMU 를 종료한다.



## Conclusion

이 Lab에서는 아래를 수행하는 방법을 배웠다:

- ARM Cortex-A9 MPcore 임베디드 리눅스 애플리케이션 만들기
- 크로스 컴파일로 애플리케이션 개발
- QEMU 에서 응용 프로그램 개발
- 시스템 디버거를 사용하여 응용 프로그램 디버깅

시스템 디버거를 사용하여 응용 프로그램을 디버깅 할 수는 있지만  
응용 프로그램에서 필요할 때마다 정보를 출력하거나 로깅하는 것은 추적 문제에 있어 매우 중요하다.

## Completed Solution

솔루션을 실행하려면 labsolution/lab3/SDCard 디렉토리의 BOOT.bin 을 SD 카드에 복사한다.  
SD 카드를 Zybo 에 넣는다.  
SD 카드 부팅 모드에서 Zybo 를 설정한다.  
이더넷 케이블을 사용하여 Zybo 를 Host 컴퓨터에 연결한다.

아래 명령을 실행하여 Host 에서 DHCP 서버를 시작한다.

```
[host]$ sudo service isc-dhcp-server restart
```

labsolution/lab3/tftpboot 디렉토리의 image.ub 파일을 /tftpboot 디렉토리로 복사한다.

보드의 전원을 켜다.  
터미널 세션을 설정한다.  
자동 부팅 메시지가 표시되면 부팅 프로세스를 중단한다.  
Target Board 터미널 창에서 아래 명령을 사용하여 server ip 주소를 설정한다.

```
#set serverip 192.169.1.1
```

netboot 명령을 실행한다.

```
#run netboot
```

시스템에 로그인하고 Lab 을 테스트한다.

# References

<https://www.xilinx.com/support/university/vivado/vivado-workshops/Vivado-embedded-linux-zynq.html>