

AM5728

Kernel Compile

menuconfig 추가, SD카드에 올리기까지

김현

먼저 기존의 TI 가 제공하고 있는 루트 파일 시스템을 활용할 필요가 있다.
그러므로 임시 파일 시스템을 저장할 공간을 아래와 같이 만들어두도록 한다.
향후 SD Card 에 구울때 파일 시스템으로 이 위치를 지정해주면 된다.

```
kwrg@kwrg-Samsung-DeskTop-System: ~  
kwrg@kwrg-Samsung-DeskTop-System:~$ mkdir tmp_fs
```

TI SDK 위치에서 filesystem 쪽으로 접근한다.

```
kwrg@kwrg-Samsung-DeskTop-System: ~/ti-processor-sdk-linux-am57xx-evm-03.02.00.05/  
kwrg@kwrg-Samsung-DeskTop-System:~/ti-processor-sdk-linux-am57xx-evm-03.02.00.05/  
filesystem$ ls  
arago-base-tisdk-image-am57xx-evm.tar.xz  tisdk-rootfs-image-am57xx-evm.tar.xz  
kwrg@kwrg-Samsung-DeskTop-System:~/ti-processor-sdk-linux-am57xx-evm-03.02.00.05/  
filesystem$
```

우선 파일 시스템을 복사해둔다.

```
kwrg@kwrg-Samsung-DeskTop-System:~/ti-processor-sdk-linux-am57xx-evm-03.02.00.05  
/filesystem$ cp tisdsk-rootfs-image-am57xx-evm.tar.xz ~/tmp_fs
```

복사한 녀석을 해당 위치에서 압축 해제한다.

```
kwrg@kwrg-Samsung-DeskTop-System:~/ti-processor-sdk-linux-am57xx-evm-03.02.00.05  
/filesystem$ cd ~/tmp_fs  
kwrg@kwrg-Samsung-DeskTop-System:~/tmp_fs$ tar -xf tisdsk-rootfs-image-am57xx-evm  
.tar.xz
```

모두 완료하였다면 지운다.

```
kwrg@kwrg-Samsung-DeskTop-System:~/ti-processor-sdk-linux-am57xx-evm-03.02.00.05  
/filesystem$ cd ~/tmp_fs  
kwrg@kwrg-Samsung-DeskTop-System:~/tmp_fs$ rm -rf tisdsk-rootfs-image-am57xx-evm.  
tar.xz
```

이 구문을 작성해야 환경변수 설정 오류가 나지 않는다.

```
koitt@koitt-Z20NH-AS51B1U: ~  
koitt@koitt-Z20NH-AS51B1U:~$ export TOOLCHAIN_PATH=~/.ti-processor-sdk-linux-am57xx-evm-03.02.00.05/linux-devkit/sysroots/x86_64-arago-linux/usr/bin/  
koitt@koitt-Z20NH-AS51B1U:~$ export CROSS_COMPILE=arm-linux-gnueabihf-  
koitt@koitt-Z20NH-AS51B1U:~$ export ARCH=arm  
koitt@koitt-Z20NH-AS51B1U:~$ [ "$TOOLCHAIN_PATH" != "DEFAULT" ] && export PATH=$TOOLCHAIN_PATH:$PATH  
koitt@koitt-Z20NH-AS51B1U:~$
```

컴파일을 할 위치로 간다.

```
koitt@koitt-Z20NH-AS51B1U:~$ cd ~/.ti-processor-sdk-linux-am57xx-evm-03.02.00.05/board-support/linux-4.4.32+gitAUTOINC+adde2ca9f8-gadde2ca9f8/
```

기존에 컴파일 되어 있는 정보를 깔끔하게 모두 정리한다.

```
kwrg@kwrg-Samsung-DeskTop-System:~/ti-processor-sdk-linux-am57xx-evm-03.02.00.05  
/board-support/linux-4.4.32+gitAUTOINC+adde2ca9f8-gadde2ca9f8$ make ARCH=arm CRO  
SS_COMPILE=arm-linux-gnueabihf- distclean
```

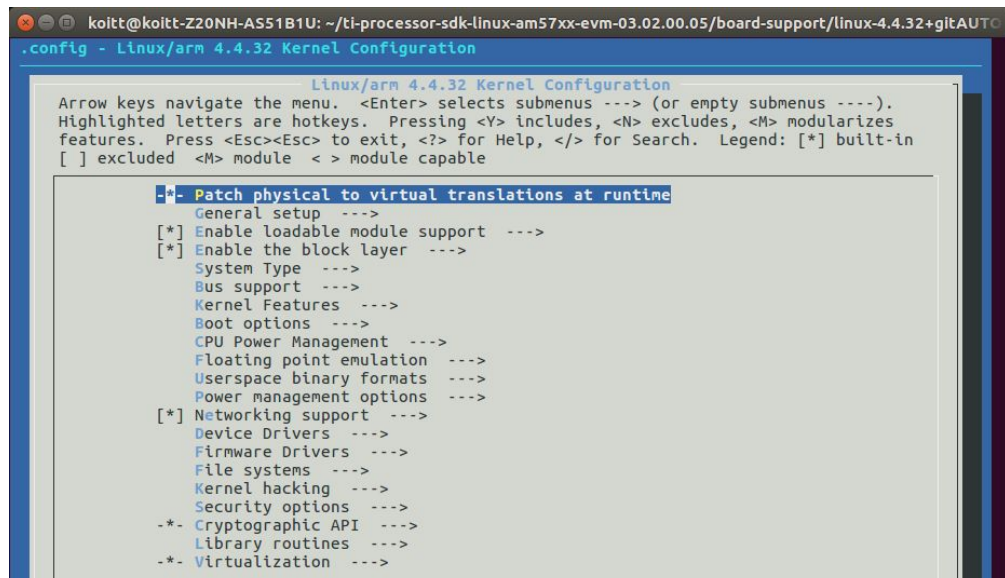
TI SDK AM57XX EVM 에 적절한 형태로 설정을 한다.

```
kwrg@kwrg-Samsung-DeskTop-System:~/ti-processor-sdk-linux-am57xx-evm-03.02.00.05  
/board-support/linux-4.4.32+gitAUTOINC+adde2ca9f8-gadde2ca9f8$ make ARCH=arm CRO  
SS_COMPILE=arm-linux-gnueabihf- tisdsk_am57xx-evm_defconfig
```

커널 설정 변경을 위해 menuconfig를 한다.

```
koitt@koitt-Z20NH-AS51B1U:~/ti-processor-sdk-linux-am57xx-evm-03.02.00.05/board-support/linux-4.4.32+gitAUTOINC+adde2ca9f8-gadde2ca9f8$ make menuconfig
```

menuconfig 메뉴에 들어가서 각자에게 맞는 설정을 한다.



```
koitt@koitt-Z20NH-AS51B1U: ~/ti-processor-sdk-linux-am57xx-evm-03.02.00.05/board-support/linux-4.4.32+gitAUTOINC+adde2ca9f8-gadde2ca9f8$ make menuconfig
.config - Linux/arm 4.4.32 Kernel Configuration

Linux/arm 4.4.32 Kernel Configuration
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty submenu ----).
Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes
features. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in
[ ] excluded <M> module < > module capable

[*] Patch physical to virtual translations at runtime
  General setup --->
  [*] Enable loadable module support --->
  [*] Enable the block layer --->
  System Type --->
  Bus support --->
  Kernel Features --->
  Boot options --->
  CPU Power Management --->
  Floating point emulation --->
  Userspace binary formats --->
  Power management options --->
  [*] Networking support --->
  Device Drivers --->
  Firmware Drivers --->
  File systems --->
  Kernel hacking --->
  Security options --->
  *- Cryptographic API --->
  Library routines --->
  *- Virtualization --->
```

menuconfig를 사용하지 않고 config파일을 직접 수정할 수도 있다.

```
koitt@koitt-Z20NH-AS51B1U:~/ti-processor-sdk-linux-am57xx-evm-03.02.00.05/board-support/linux-4.4.32+gitAUTOINC+adde2ca9f8-gadde2ca9f8$ vi .config
```

config 파일을 볼 수 있다. 수정도 가능하다.

```
CONFIG_FIX_EARLYCON_MEM=y
CONFIG_GENERIC_HWEIGHT=y
CONFIG_GENERIC_CALIBRATE_DELAY=y
CONFIG_ZONE_DMA=y
CONFIG_NEED_DMA_MAP_STATE=y
CONFIG_ARCH_SUPPORTS_UPROBES=y
CONFIG_VECTORS_BASE=0xffff0000
CONFIG_ARM_PATCH_PHYS_VIRT=y
CONFIG_GENERIC_BUG=y
CONFIG_PGTABLE_LEVELS=3
CONFIG_DEFCONFIG_LIST="/lib/modules/$UNAME_RELEASE/.config"
CONFIG_IRQ_WORK=y
CONFIG_BUILDTIME_EXTABLE_SORT=y

#
# General setup
#
CONFIG_INIT_ENV_ARG_LIMIT=32
CONFIG_CROSS_COMPILE=""
# CONFIG_COMPILE_TEST is not set
CONFIG_LOCALVERSION=""
CONFIG_LOCALVERSION_AUTO=y
CONFIG_HAVE_KERNEL_GZIP=y
CONFIG_HAVE_KERNEL_LZMA=y
CONFIG_HAVE_KERNEL_XZ=y
CONFIG_HAVE_KERNEL_LZO=y
CONFIG_HAVE_KERNEL_LZ4=y
# CONFIG_KERNEL_GZIP is not set
CONFIG_KERNEL_LZMA=y
# CONFIG_KERNEL_XZ is not set
```

=m -> 모듈

=y -> built in

is not set -> 해제

실제 Kernel Image 를 컴파일 한다.

```
kwrg@kwrg-Samsung-DeskTop-System:~/ti-processor-sdk-linux-am57xx-evm-03.02.00.05  
/board-support/linux-4.4.32+gitAUTOINC+adde2ca9f8-gadde2ca9f8$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- zImage
```

Kernel 의 Device Tree 부분을 컴파일 한다.

```
kwrg@kwrg-Samsung-DeskTop-System:~/ti-processor-sdk-linux-am57xx-evm-03.02.00.05  
/board-support/linux-4.4.32+gitAUTOINC+adde2ca9f8-gadde2ca9f8$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- am57xx-evm-reva3.dtb
```

reva3 과 일반을 모두 해두도록 한다.

```
kwrg@kwrg-Samsung-DeskTop-System:~/ti-processor-sdk-linux-am57xx-evm-03.02.00.05  
/board-support/linux-4.4.32+gitAUTOINC+adde2ca9f8-gadde2ca9f8$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- am57xx-evm.dtb
```


Linux Device Driver 에 해당하는 Module 을 컴파일 하기 시작한다.

```
kwrg@kwrg-Samsung-DeskTop-System:~/ti-processor-sdk-linux-am57xx-evm-03.02.00.05  
/board-support/linux-4.4.32+gitAUTOINC+adde2ca9f8-gadde2ca9f8$ make ARCH=arm CRO  
SS_COMPILE=arm-linux-gnueabihf- modules
```

앞서서 미리 작업해놔던 임시 파일 시스템쪽에 Module 을 설치한다.

```
kwrg@kwrg-Samsung-DeskTop-System:~/ti-processor-sdk-linux-am57xx-evm-03.02.00.05  
/board-support/linux-4.4.32+gitAUTOINC+adde2ca9f8-gadde2ca9f8$ make ARCH=arm INS  
TALL_MOD_PATH=/home/kwrg/tmp_fs modules_install
```

U-Boot 컴파일을 수행하기 위해 Device Tree Compiler 를 설치하도록 한다.

```
kwrg@kwrg-Samsung-DeskTop-System:~/ti-processor-sdk-linux-am57xx-evm-03.02.00.05  
/board-support/linux-4.4.32+gitAUTOINC+adde2ca9f8-gadde2ca9f8$ sudo apt-get inst  
all device-tree-compiler
```

이제 U-Boot 관련 부분을 작업하도록 위치를 이동시키도록 한다.

```
kwrg@kwrg-Samsung-DeskTop-System:~/ti-processor-sdk-linux-am57xx-evm-03.02.00.05
/board-support/linux-4.4.32+gitAUTOINC+adde2ca9f8-gadde2ca9f8$ cd ..
kwrg@kwrg-Samsung-DeskTop-System:~/ti-processor-sdk-linux-am57xx-evm-03.02.00.05
/board-support$ ls
extra-drivers
linux-4.4.32+gitAUTOINC+adde2ca9f8-gadde2ca9f8
prebuilt-images
u-boot-2016.05+gitAUTOINC+6c5519b6fc-g6c5519b6fc
kwrg@kwrg-Samsung-DeskTop-System:~/ti-processor-sdk-linux-am57xx-evm-03.02.00.05
/board-support$ cd u-boot-2016.05+gitAUTOINC+6c5519b6fc-g6c5519b6fc/
```

U-Boot 에서도 기존의 정보를 깔끔하게 정리한다.

```
kwrg@kwrg-Samsung-DeskTop-System:~/ti-processor-sdk-linux-am57xx-evm-03.02.00.05
/board-support/u-boot-2016.05+gitAUTOINC+6c5519b6fc-g6c5519b6fc$ make ARCH=arm C
ROSS_COMPILE=arm-linux-gnueabihf- distclean
```

우리 Target EVM 에 적절하게 크로스 컴파일을 수행한다.

```
kwrg@kwrg-Samsung-DeskTop-System:~/ti-processor-sdk-linux-am57xx-evm-03.02.00.05  
/board-support/u-boot-2016.05+gitAUTOINC+6c5519b6fc-g6c5519b6fc$ make ARCH=arm C  
ROSS_COMPILE=arm-linux-gnueabihf- O=am57xx_evm am57xx_evm_defconfig
```

```
kwrg@kwrg-Samsung-DeskTop-System:~/ti-processor-sdk-linux-am57xx-evm-03.02.00.05  
/board-support/u-boot-2016.05+gitAUTOINC+6c5519b6fc-g6c5519b6fc$ make ARCH=arm C  
ROSS_COMPILE=arm-linux-gnueabihf- O=am57xx_evm
```

ls를 눌러 u-boot.img 파일과 MLO파일을 확인한다.

```
koitt@koitt-Z20NH-AS51B1U:~/ti-processor-sdk-linux-am57xx-evm-03.02.00.05/board-support/u-boot-2016.0  
5+gitAUTOINC+6c5519b6fc-g6c5519b6fc$ ls
```

Kbuild	README	common	examples	snapshot.commit	u-boot-nodtb.bin	u-boot.srec
Kconfig	System.map	config.mk	fs	spl	u-boot.bin	u-boot.sym
Licenses	TISDK-README	configs	include	test	u-boot.cfg	
MAINTAINERS	api	disk	lib	tools	u-boot.dtb	
MAKEALL	arch	doc	net	u-boot	u-boot.img	
MLO	board	drivers	post	u-boot-dtb.bin	u-boot.lds	
Makefile	cmd	dts	scripts	u-boot-dtb.img	u-boot.map	

u-boot.img 파일과 MLO파일을 boot로 만들어 놓은 폴더에 복사한다.

```
koitt@koitt-Z20NH-AS51B1U:~/ti-processor-sdk-linux-am57xx-evm-03.02.00.05/board-support/u-boot-2016.05+gitAUTOINC+6c5519b6fc-g6c5519b6fc$ cp u-boot.img ~/new/tmp_boot/  
koitt@koitt-Z20NH-AS51B1U:~/ti-processor-sdk-linux-am57xx-evm-03.02.00.05/board-support/u-boot-2016.05+gitAUTOINC+6c5519b6fc-g6c5519b6fc$ cp MLO ~/new/tmp_boot/  
koitt@koitt-Z20NH-AS51B1U:~/ti-processor-sdk-linux-am57xx-evm-03.02.00.05/board-support/u-boot-2016.05+gitAUTOINC+6c5519b6fc-g6c5519b6fc$
```

SD카드 만들기 위한 명령어를 작성한다.

```
koitt@koitt-Z20NH-AS51B1U:~/ti-processor-sdk-linux-am57xx-evm-03.02.00.05/board-support/u-boot-2016.05+gitAUTOINC+6c5519b6fc-g6c5519b6fc$ cd ~  
koitt@koitt-Z20NH-AS51B1U:~$ sudo ti-processor-sdk-linux-am57xx-evm-03.02.00.05/bin/create-sdcard.sh  
█
```


리눅스 커널을 올릴 SD카드를 선택한다.

```
Available Drives to write images to:
```

```
# major  minor    size  name
1:    8        48 15558144 sdd
```

```
Enter Device Number or n to exit: 1
```

SD카드를 몇 개의 부분으로 나눌지 파티션을 정한다. (보통 2이다. boot부분

```
#####
```

```
Select 2 partitions if only need boot and rootfs (most users).
Select 3 partitions if need SDK & other content on SD card. This is
usually used by device manufacturers with access to partition tarballs.
```

```
****WARNING**** continuing will erase all data on sdd
```

```
#####
```

```
Number of partitions needed [2/3] : 2
```

그 후로 y를 쳐서 진행한다.

```
#####

Partitioning Boot

#####
mkfs.fat 3.0.28 (2015-05-16)
mkfs.fat: warning - lowercase labels might not work properly with DOS or Windows
mkfs.vfat: /dev/sdd1 contains a mounted filesystem.

#####

Partitioning rootfs

#####
mke2fs 1.42.13 (17-May-2015)
/dev/sdd2 contains a ext3 file system labelled 'rootfs'
        last mounted on / on Tue Aug  1 20:16:44 2017
Proceed anyway? (y,n) y
```

```
#####

Partitioning is now done
Continue to install filesystem or select 'n' to safe exit

**Warning** Continuing will erase files any files in the partitions

#####

Would you like to continue? [y/n] : y
```

여기서 custom을 선택한다. (2를 쓴다)

```
#####  
Choose file path to install from  
  
1 ) Install pre-built images from SDK  
2 ) Enter in custom boot and rootfs file paths  
  
#####  
Choose now [1/2] : 2
```

아까 커널 컴파일 중 만들어 뒀던 boot 폴더의 경로를 넣고 y를 눌러

```
#####  
For U-boot and MLO  
  
If files are located in Tarball write complete path including the file name.  
e.x. $: /home/user/MyCustomTars/boot.tar.xz  
  
If files are located in a directory write the directory path  
e.x. $: /ti-sdk/board-support/prebuilt-images/  
  
NOTE: Not all platforms will have an MLO file and this file can  
be ignored for platforms that do not support an MLO.  
  
Update: The proper location for the kernel image and device tree  
files have moved from the boot partition to the root filesystem.  
  
#####  
Enter path for Boot Partition : /home/koitt/new/tmp_boot/  
  
Directory exists  
  
This directory contains:  
MLO u-boot.img  
  
Is this correct? [y/n] : y
```


커널 이미지와 디바이스 트리 설정은 1번으로 해준다.

```
#####  
For Kernel Image and Device Trees files  
  
What would you like to do?  
1) Reuse kernel image and device tree files found in the selected rootfs.  
2) Provide a directory that contains the kernel image and device tree files  
   to be used.  
  
#####  
Choose option 1 or 2 : 1
```

아까 커널 컴파일 중 만들어 뒀던 **rootfs** 폴더의 경로를 넣고 **y**를 눌러

```
#####  
For Rootfs partition  
  
If files are located in Tarball write complete path including the file name.  
e.x. $: /home/user/MyCustomTars/rootfs.tar.xz  
  
If files are located in a directory write the directory path  
e.x. $: /ti-sdk/targetNFS/  
  
#####  
Enter path for Rootfs Partition : /home/koitt/new/tmp_fs/  
  
This directory contains:  
bin  dev  home  lib  mnt  proc  sbin  sys  usr  www  
boot etc  include media opt  run  srv  tmp  var  
  
Is this correct? [y/n] : y
```

SD카드 만들기를 완료한다.

```
#####  
      Copying files now... will take minutes  
#####  
Copying boot partition  
MLO copied  
  
u-boot.img copied  
  
Copying rootfs System partition  
2039340 / 2024188 copied  
  
Syncing...  
  
Un-mount the partitions  
  
Remove created temp directories  
  
Operation Finished  
  
koitt@koitt-Z20NH-AS51B1U:~$
```