Armadillo-640
開発基礎セミナー

第5部 アプリケーションの作成

株式会社アットマークテクノ
http://www.atmark-techno.com/

目次

この章の概要

- エディタの使用方法
- 簡単なアプリケーションの作成（C言語編）
- 簡単なアプリケーションの作成（Python編）

Armadillo-640
アプリケーション開発環境

- **Armadillo-640のアプリケーションはArmadillo上で行います。**
  - 組み込み開発でよくあるクロス開発ではなく、セルフ開発が基本となります。

- **Armadillo-640では、Debian GNU/Linuxが動作しているため、各種言語が使用可能。**
  - C/C++
  - JAVA
  - node.js, Ruby, Python等

前章では、C/C++とPythonを使用するためのパッケージをインストールしました。

Armadillo-640
アプリケーション開発環境

- この章では、簡単なサンプルを作成し、動作させます

- 例として以下の言語でサンプルを作成します
  - C
  - Python

ファイル編集のエディタについて

- **Armadilloでは、ファイル編集はエディタを使用します。**
- **殆どのLinuxシステムに標準でviエディタがインストールされています。**
- **Windowsで一般的なエディタ(メモ帳など)とは異なり、モードを持っている事が大きな特徴です。**
  - コマンドモード：入力した文字は全てコマンドとして扱います
  - 入力モード：通常のエディタと同様に、文字の入力ができます

viエディタ基本操作

- Vi エディタ"test"というファイルを開きます。

```
root@armadillo:~# vi test
```

- 起動直後はコマンドモードです。
- 入力モードに切り替える為には以下のいずれかのキーを入力します。

入力キー	動作
i	カーソルのある位置から入力開始
a	カーソルのある後ろから入力開始

- 再度コマンドモードに戻る際は「Esc」キーを入力します。
- コマンドモード時に、「x」でカーソル上の文字、「dd」コマンドでカーソルのある行を削除できます。
- 終了したい場合は、以下のいずれかのコマンドを実行します。

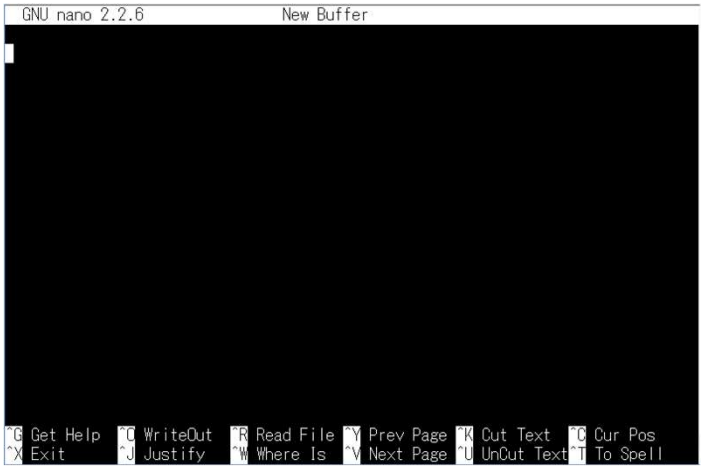
コマンド	動作
:q!	変更を保存せずに終了
:w [ファイル名]	ファイル名を指定して保存
:wq	ファイルを上書き保存して終了

【参考情報】nanoエディタの使用方法1

- 普段メモ帳等を使用している方は、viエディタは慣れるまで操作が難しいかと思います。
- メモ帳に近い感覚で使用できるnanoエディタもインストールして使用することができます。

```
root@armadillo:~# apt-get install nano
```

- ご自身が使いやすいエディタをご利用ください。



【参考情報】 nanoエディタの使用方法2

- nanoエディタでtestというファイルを作成してみます

```
root@armadillo:~# nano test
```

- エディタ画面の末尾の2行はヘルプです。

```
^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
^X Exit      ^R Read File ^¥ Replace   ^U Uncut Text^T To Spell  ^_ Go To Line
```

- ^X (Ctrl + x キー)で終了を選択します。
- Saveの確認では、保存したい場合は y キーを選択します。

```
Save modified buffer?  (Answering "No" will DISCARD changes.)
 Y Yes
 N No           ^C Cancel
```

- ファイル名の確認では、Enter　キーを入力します。

```
File Name to Write: temp
```

以上で、最初に指定した"test"というファイルが作成されます。

簡易的なプログラムの作成

簡易的なプログラム

- ここでは、Armadillo-640の赤LEDを点灯するプログラムを作成してみます。
- まずは、シェルスクリプトで赤LEDの点灯/消灯を確認します。
- 赤LEDの点灯：下記のコマンドを実行します。

```
root@armadillo:~# echo 1 > /sys/class/leds/red/brightness
```

- 赤LEDの消灯：下記のコマンドを実行します。

```
root@armadillo:~# echo 0 > /sys/class/leds/red/brightness
```

- 解説：LEDはクラスで定義されています。（/sys/class）
  leds/red：赤LEDを意味します。
  brightness：0で消灯、1で点灯します。

echoコマンドで、brightnessに0,1を書き込んで、点灯/消灯します。

赤LEDを点灯（C言語編1）

- ”redc.c”というファイル名で、nanoエディタを開きます。

```
root@armadillo:~# nano redc.c
```

- 下記のように編集して保存して終了します。

```
#include <stdio.h>

int main(void)
{
  FILE *fp;

  fp = fopen("/sys/class/leds/red/brightness", "w");
  fprintf(fp, "1");
  fclose(fp);

  return 0;
}
```

赤LEDを点灯（C言語編2）

- ”redc.c”をビルドします。(実行ファイル名は、”redc”)

```
root@armadillo:~# gcc redc.c -o redc
```

（前章で”build-essential”をインストールすることによりビルドできるようになってます。）

- ”redc”を実行します。

```
root@armadillo:~# ./redc
```

- 赤LEDが点灯することを確認します。

元々、赤LEDが点灯している場合は、

```
root@armadillo:~# echo 0 > /sys/class/leds/red/brightness
```

で、赤LEDを消灯してから、確認します。

赤LEDを点灯（Python編1）

- ”redp.py”というファイル名で、nanoエディタを開きます。

```
root@armadillo:~# nano redp.py
```

- 下記のように編集して保存して終了します。

```
f = open("/sys/class/leds/red/brightness", "w")
print >> f, "1"
f.close()
```

赤LEDを点灯（Python編2）

- ”redp.py”を実行します。

```
root@armadillo:~# python redp.py
```

（前章で”python”をインストールすることにより実行できるようになってます。）

- 赤LEDが点灯することを確認します。

  元々、赤LEDが点灯している場合は、

```
root@armadillo:~# echo 0 > /sys/class/leds/red/brightness
```

  で、赤LEDを消灯してから、確認します。

参考）C言語：Makefileによるビルド1

- C言語のソースコードをビルドするとき、”Makefile”ファイルにソースコード名、実行ファイル名、ビルドオプションなどのルールを記述することで、makeコマンドでビルドできます。

- 先ほど作成したした実行ファイル”redc”を削除します。

```
root@armadillo:~# make clean
```

- makeコマンドでビルドします。

```
root@armadillo:~# make
gcc -Wall -Wextra -O2 -c -o redc.o redc.c
gcc  redc.o  -o redc
```

- Makefileのルールに従って、ビルドされ、”redc”という実行ファイルが生成されます。

参考）C言語：Makefileによるビルド2

- redc.c用のMakefileは、下記のようになってます。

```
CC = gcc
CFLAGS = -Wall -Wextra -O2
LDFLAGS =

TARGET = redc

all: $(TARGET)

redc: redc.o
	$(CC) $(LDFLAGS) $^ $(LDLIBS) -o $@

clean:
	$(RM) *~ *.o $(TARGET)

%.o: %.c
	$(CC) $(CFLAGS) -c -o $@ $<
```

インデントは必ずTABで行います。