1. 嵌入式linux应用程序怎样编译和运行

在主机上使用arm-linux-gcc编译生成目标板嵌入式linux可运行的ELF二进制文件，在目标板上使用./Sample（文件名）即可运行

1. GDB调试软件的常用命令
2. 启动GDB调试程序

GDB Sample（可执行ELF文件）

1. 退出GDB调试

在GDB中输入quit(q)

1. 命令行传参

set argc(参数个数) \_\_ \_\_ \_\_

1. GDB查看源代码

list(l)

l 行号/函数名

1. GDB设置断点break(b)

普通断点：

b 代码行号/函数名

条件断点：

b 行号 if $var == var

查看设置过的断点

Info(i) b

删除断点

del(d) 断点编号

无效化断点

disable(dis) 断点编号

有效化断点

enable(ena) 断点编号

1. 单步步过

next(n)

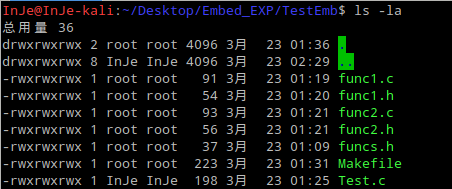
1. 单步步入

step(s)

1. 函数跳出

finish

1. 试编写一个makefile文件。一个工程由3个源文件组成，分别是main.c func1.c func2.c，其中func1.c定义了float adder(float x, float y)函数，func2.c中定义了一个函数float minus(float x, float y)，主函数中有代码用于测试func1.c和func2.c中的两个函数是否正确。写出工程，并调试通过。

//func1.h

//Add two float numbers

float adder(float x, float y);

//func1.c

#include "func1.h"

//Add two float numbers

float adder(float x, float y)

{

return x + y;

}

//func2.h

//Minus two float numbers

float minus(float x, float y);

//func2.c

#include "func2.h"

//Minus two float numbers

float minus(float x, float y)

{

return x - y;

}

//funcs.h

#include "func1.h"

#include "func2.h"

//Test.c

include<unistd.h>

#include<stdio.h>

#include "funcs.h"

int main(void)

{

//Test 1.2+2.3 == 3.5

printf("%f\n", adder(1.2,2.3));

//Test 5.3-2.1 == 3.2

printf("%f\n", minus(5.3,2.1));

return 0;

}

//Makefile

CC=gcc

ADDSUB:Test.o func1.o func2.o

$(CC) -o ADDSUB Test.o func1.o func2.o

Test.o:Test.c funcs.h

$(CC) -c Test.c

func1.o:func1.c func1.h

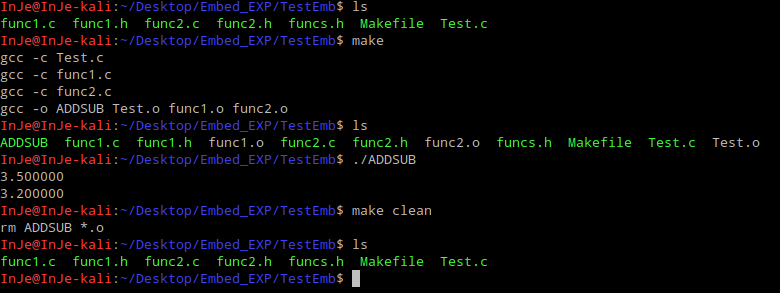
$(CC) -c func1.c

func2.o:func2.c func2.h

$(CC) -c func2.c

clean:

rm ADDSUB \*.o



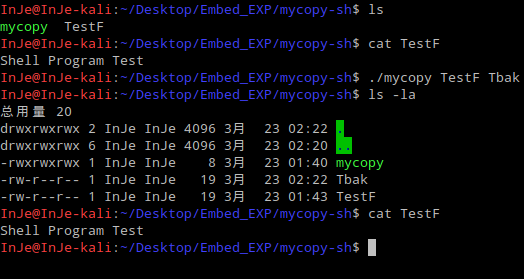
1. 对Linux下C编程的文件操作，有哪两种主要的方法

ANSI国际标准的f系列函数（fopen、fclose、fgetc、fwrite等）

POSIX可移植操作系统接口，不局限于具体操作系统，只要是支持POSIX标准的操作系统就能使用，有open、close、read、write等函数

1. Linux系统中，对文件进行C编程操作，有哪些步骤
2. 新建源程序文件
3. 编写源文件（包括引用必要的头文件，stdio.h unistd.h等）
4. 保存源文件并用gcc编译生成可执行的ELF二进制文件
5. 编写一个程序，利用linux的文件操作，完成文件拷贝功能，源文件名为mycopy，程序在命令行下运行，方法：mycopy sourcefile targetfile

//Shell

cp $1 $2

//C-POSIX

#include<sys/types.h>

#include<sys/stat.h>

#include<unistd.h>

#include<stdio.h>

#include<stdlib.h>

#include<fcntl.h>

#define BufferSize 1024

int main(int argc, char\* args[])

{

int FDread = 0, FDwrite = 0;

char temp[BufferSize] = { 0 };

int len;

if (argc != 3)

{

printf("mycopy [src] [dest]\n");

}

else

{

FDread = open(args[1], O\_RDONLY);

FDwrite = open(args[2], O\_CREAT | O\_RDWR, 00644);

while (len = read(FDread, temp, BufferSize))

{

write(FDwrite, temp, len);

}

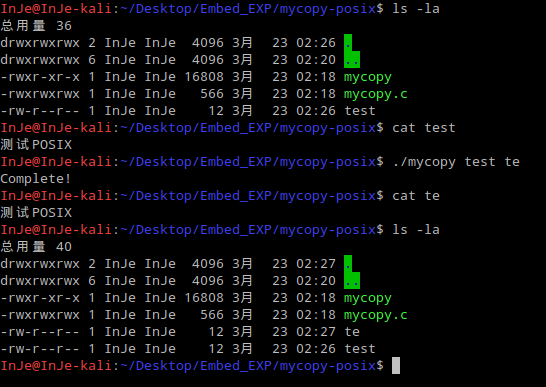
close(FDread);

close(FDwrite);

printf("Complete!\n");

}

return 0;

}

//C-ANSI

#include<stdio.h>

#include<stdlib.h>

#define BufferSize 1024

int main(int argc, char\* args[])

{

FILE \*pFread = NULL, \*pFwrite = NULL;

char temp[BufferSize] = { 0 };

int len;

if (argc != 3)

{

printf("mycopy [src] [dest]\n");

}

else

{

pFread = fopen(args[1], "rb");

pFwrite = fopen(args[2], "wb");

while (len = fread(temp, 1, BufferSize, pFread))

{

fwrite(temp, 1, len, pFwrite);

}

fclose(pFread);

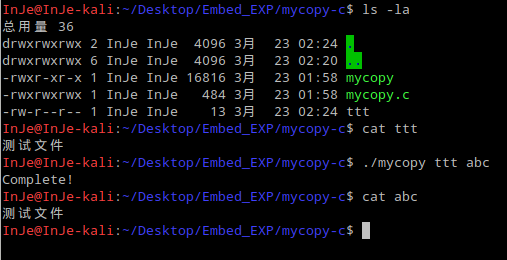
fclose(pFwrite);

printf("Complete!\n");

}

return 0;

}



1. 编写一个程序，利用linux的文件操作，完成文件比较功能，源文件名为mycomp，程序在命令行下运行，方法：mycomp sourcefile targetfile，若两个文件相同，显示：两个文件相同；否则指出两个文件不同的原因（先判断文件长度，再判断第一个不同字节的位置）

//C-ANSI

#include<stdio.h>

#include<stdlib.h>

#define BufferSize 8192

//Compare the two buffers

long DataComp(char \*s1, char \*s2, long len)

{

//Offset to the begin

long ret = 0;

while (len--)

{

if (\*s1 != \*s2)

break;

s1++, s2++, ret++;

}

return ret;

}

int main(int argc, char\* args[])

{

FILE \*pF1= NULL, \*pF2 = NULL;

char temp1[BufferSize] = { 0 }, temp2[BufferSize] = { 0 };

int DiffFlag = 0;

int len, offset, count = 0;

if (argc != 3)

{

printf("mycomp [src1] [src2]\n");

}

else

{

pF1 = fopen(args[1], "rb");

pF2 = fopen(args[2], "rb");

fseek(pF1, 0, SEEK\_END);

fseek(pF2, 0, SEEK\_END);

if (ftell(pF1) != ftell(pF2))

{

printf("Size doesn't match.\n");

}

else

{

fseek(pF1, 0, SEEK\_SET);

fseek(pF2, 0, SEEK\_SET);

while (len = fread(temp1, 1, BufferSize, pF1))

{

fread(temp2, 1, BufferSize, pF2);

//Find difference

if ((offset = DataComp(temp1, temp2, len)) < len)

{

DiffFlag = 1;

printf("Same Size, but different data.\nOffset:%ld\n", count \* BufferSize + offset);

break;

}

count++;//Buffer Page++

}

//The same

if (!DiffFlag)

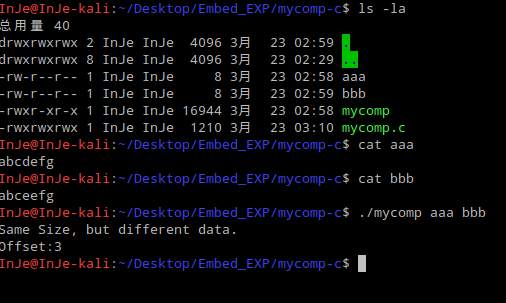
printf("Same file.\n");

}

fclose(pF1);

fclose(pF2);

}



return 0;

}

//C-POSIX

#include<sys/stat.h>

#include<sys/types.h>

#include<unistd.h>

#include<stdio.h>

#include<stdlib.h>

#include<fcntl.h>

#define BufferSize 8192

//Compare the two buffers

long DataComp(char \*s1, char \*s2, long len)

{

//Offset to the begin

long ret = 0;

while (len--)

{

if (\*s1 != \*s2)

break;

s1++, s2++, ret++;

}

return ret;

}

int main(int argc, char\* args[])

{

int FD1= 0, FD2 = 0;

char temp1[BufferSize] = { 0 }, temp2[BufferSize] = { 0 };

int DiffFlag = 0;

int len, offset, count = 0;

if (argc != 3)

{

printf("mycomp [src1] [src2]\n");

}

else

{

FD1 = open(args[1], O\_RDONLY);

FD2 = open(args[2], O\_RDONLY);

if (lseek(FD1, 0, SEEK\_END) != lseek(FD2, 0, SEEK\_END))

{

printf("Size doesn't match.\n");

}

else

{

lseek(FD1, 0, SEEK\_SET);

lseek(FD2, 0, SEEK\_SET);

while (len = read(FD1, temp1, BufferSize))

{

read(FD2, temp2, BufferSize);

//Find difference

if ((offset = DataComp(temp1, temp2, len)) < len)

{

DiffFlag = 1;

printf("Same Size, but different data.\nOffset:%ld\n", count \* BufferSize + offset);

break;

}

count++;//Buffer Page++

}

//The same

if (!DiffFlag)

printf("Same file.\n");

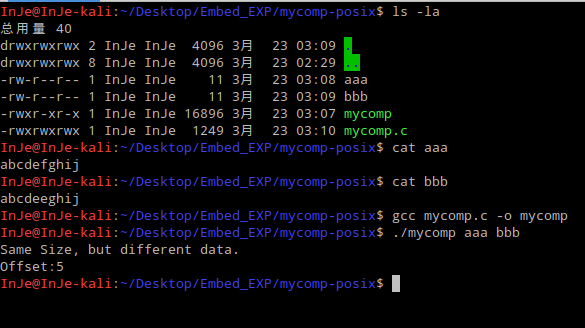
}

close(FD1);

close(FD2);

}

return 0;

}

1. 标准库文件操作与POSIX文件操作有什么不同

C的POSIX文件操作库是C标准库为了POSIX子系统提供的超集，它与ANSI C标准在同时开发。POSIX比标准库文件多包含一些额外的功能。POSIX的文件操作函数与C标准库文件操作的函数不同，POSIX的文件操作接口主要是通过C库（libc）实现的，是在C标准库上的一层抽象曾以实现在不同操作系统间的兼容性。

举例：在C标准库中打开文件是fopen()，而在POSIX中是open()；在C标准库中关闭文件是fclose()，而在POSIX中是close()等等。

