

Forum:[Hack Forums](#)

Thread: [Can we assign function return value to function pointer?](#)

[Rn] 7s^2 - July 13, 2013, 10:05 pm

"One common mistake is to do this:"

```
Code
int foo()
{
}

int goo()
{
}

int main()
{
    int (*pFoo)() = foo; // pFoo points to function foo()
    pFoo = goo; // pFoo now points to function goo()
    return 0;
}
```

Code 1

```
pFoo = goo();
```

"This would actually assign the return value from a call to function goo() to pFoo, which isn't what we want. We want pFoo to be assigned to function goo, not the return value from goo(). So no parenthesis are needed."

- Alex from [learncpp.com](#)

So I tried to do this and got compiler error and this completely makes sense because I am trying to assign int value to int function pointer.

But Alex did not talk about any compiler error so what exactly he was trying to explain here. Can someone please explain?

g4143 - July 14, 2013, 1:03 am

Can we assign function return value to function pointer?

Sure, if the function returns a pointer to a function.

[Rn] 7s^2 - July 14, 2013, 3:06 am

Learning C++ or any other programming language requires thinking wide and keeping all the stuff learned into mind. It was so simple but I didn't think that. Thanks a lot for answer.

[Rn] 7s^2 - September 10, 2013, 9:44 pm

2 months passed and I am kind of lost in this.

Can you please tell me the syntax of making a function which returns a function pointer

I tried doing:

```
Code (*int) a()
{
    return pointerFunction;
}
but it is not working.
```

6577657534735 - *September 10, 2013, 9:49 pm*

Never done this before but after a bit of quick research I'll take a guess

```
fptr a()
{
    return &pointerFunction;
}
```

g4143 - *September 11, 2013, 3:13 am*

Here's a simple example of what your looking for.

```
Code #include <stdio.h>

typedef void (*pfunc)(void);

void display_hello()
{
    fputs("Hello, World!\n", stdout);
}

pfunc try_this()
{
    return display_hello;
}

int main(int argc, char**argv)
{
    pfunc ans = try_this();
    fputs("Now call ans->", stdout);
    ans();
    return 0;
}
```

Or you could try something a little different...

```
Code #include <stdio.h>

typedef int (*pfunc)(int);

int double_it(int x)
{
    return x * 2;
}

int sqr_it(int x)
{
    return x * x;
}
```

```
}
```

```
pfunc get_func()
{
    int choice = 0;
    fputs("Enter choice (1) double it (2) square it->", stdout);
    fscanf(stdin, "%d", &choice);

    if (choice < 2)
        return double_it;
    return sqr_it;
}
```

```
int main(int argc, char**argv)
{
    int val = 0;
    fputs("Enter a value to calculate->", stdout);
    fscanf(stdin, "%d", &val);

    fprintf(stdout, "ans->%d\n", (get_func())(val));
    return 0;
}
```