DesignNews

Expert C Techniques to Master Baremetal Programming

DAY 1 : Mastering Function Pointers for Baremetal Systems

Sponsored by

DigiKey

©2025 Beningo Embedded Group, LLC. All Rights Reserved.

Webinar Logistics

- Turn on your system sound to hear the streaming presentation.
- If you have technical problems, click “Help” or submit a question asking for assistance.
- Participate in ‘Group Chat’ by maximizing the chat widget in your dock.

©2025 Beningo Embedded Group, LLC. All Rights Reserved.

THE SPEAKER

Jacob Beningo

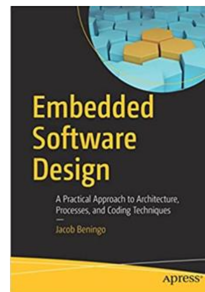
Jacob@beningo.com

jacobbeningo

Beningo Embedded Group – CEO / Founder

Focus: Software Architecture, Processes, and Dev Skills

At Beningo Embedded Group, we believe everyone deserves the skills to confidently advance their careers, meet deadlines, and deliver quality embedded systems. We provide modern strategies, insights, and hands-on training to equip developers and teams with the tools they need to succeed.



Visit www.beningo.com to learn more

©2025 Beningo Embedded Group, LLC. All Rights Reserved.

01

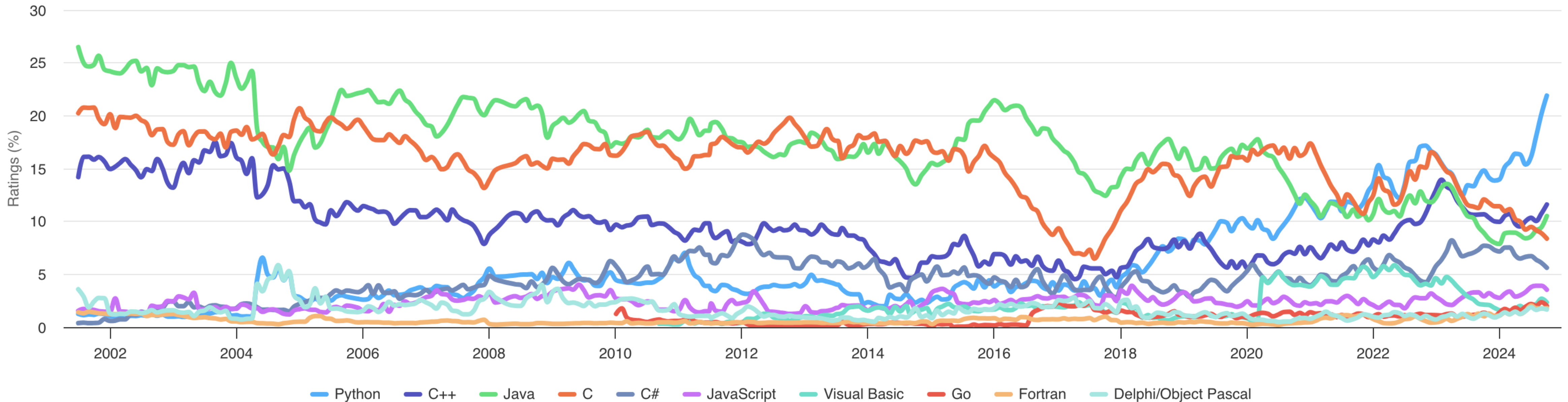
Embedded Programming Languages

- "You can't allow tradition to get in the way of innovation. There's a need to respect the past, but it's a mistake to revere your past."
  - Bob Iger

©2025 Beningo Embedded Group, LLC. All Rights Reserved.

Embedded Programming Languages

General Language Popularity



©2025 Beningo Embedded Group, LLC. All Rights Reserved.

CEC Continuing Education Center | BENINGO EMBEDDED GROUP | DigiKey

Embedded Programming Languages

Embedded Software Languages

Most Popular Embedded

- C (60 - 70%)
- C++ (20% - 25%)
- Python (<5%)
- Assembly
- Other

Note: 13-14% of Rust Developers are developing bare-metal embedded systems! Source

Oct 2024	Oct 2023	Change	Programming Language	Ratings	Change
1	1		Python	21.90%	+7.08%
2	3	^	C++	11.60%	+0.93%
3	4	^	Java	10.51%	+1.59%
4	2	v	C	8.38%	-3.70%
5	5		C#	5.62%	-2.09%
6	6		JavaScript	3.54%	+0.64%
7	7		Visual Basic	2.35%	+0.22%
8	11	^	Go	2.02%	+0.65%
9	16	^^	Fortran	1.80%	+0.78%
10	13	^	Delphi/Object Pascal	1.68%	+0.38%
13	20	^^	Rust	1.45%	+0.53%
16	10	vv	Assembly language	1.13%	-0.51%

©2025 Beningo Embedded Group, LLC. All Rights Reserved.

Embedded Programming Languages

Why is C so popular?

C is a procedural programming language that is too good at what it does.

- Developed in the early 1970s by Dennis Ritchie at Bell Labs for the UNIX operating system
- Known for its minimal runtime and high performance, making it ideal for low-level systems programming
- Provides direct access to memory and hardware through pointers and manual memory management
- Emphasizes simplicity, giving developers fine-grained control over system resources
- Lacks built-in mechanisms for safety, requiring disciplined coding to avoid memory leaks, undefined behavior, and data races
- The foundation of many modern programming languages and operating systems, including Linux and Windows
- Extensive ecosystem of libraries and tools, with wide compiler support across platforms

©2025 Beningo Embedded Group, LLC. All Rights Reserved.

Audience POLL Question

What is your primary development language?

a) C
b) C++
c) Python / MicroPython
d) Rust
e) Other

©2025 Beningo Embedded Group, LLC. All Rights Reserved.

CEC Continuing Education Center

BENINGO EMBEDDED GROUP

DigiKey

Pointers

02

©2025 Beningo Embedded Group, LLC. All Rights Reserved.

Pointers

Definition

In C, a pointer is a variable that stores the memory address of another variable.

Pointers allow for direct access and manipulation of memory, enabling powerful operations like dynamic memory allocation, efficient array handling, and passing large structures to functions without copying their contents.

Pointer Declaration

```
#include <stdio.h>

int main() {
  int x = 10;
  int *ptr = &x; // Pointer to x

  printf("Value of x: %d\n", *ptr); // Dereferencing
  printf("Address of x: %p\n", ptr); // Pointer value

  return 0;
}
```

©2025 Beningo Embedded Group, LLC. All Rights Reserved.

Pointers

Syntax

Declaring a variable:

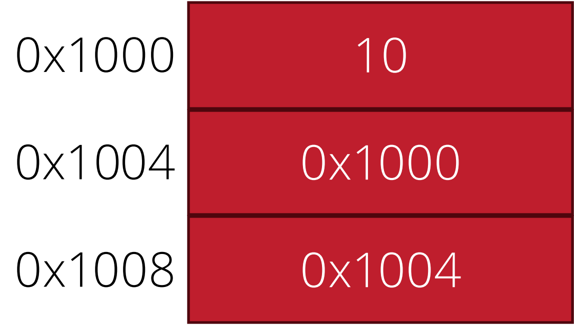
```
int x = 10;
```

Define a pointer:

```
int *ptr = &x;
```

Define a pointer to a pointer :

```
int **ptrPtr = &ptr;
```



Dereference Pointer:

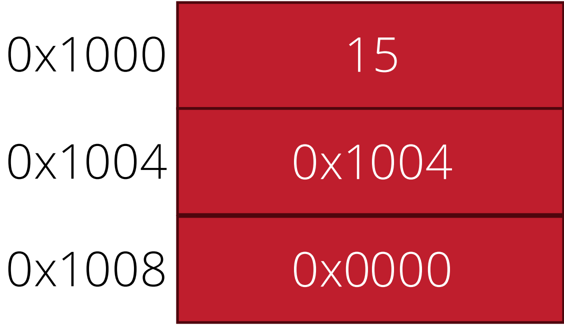
```
*ptr = 15;
```

Increment Pointer:

```
ptr++;
```

NULL Pointer

```
ptrPtr = NULL;
```



©2025 Beningo Embedded Group, LLC. All Rights Reserved.

CEC Continuing Education Center | BENINGO EMBEDDED GROUP | DigiKey

Pointers

Function Pointers

A function pointer in C is a pointer that stores the address of a function. This allows the function to be called indirectly through the pointer. Function pointers enable flexible and dynamic behavior, such as implementing callbacks, state machines, or selecting algorithms at runtime.

```c
#include <stdio.h>

// Function to be pointed to
void greet(int times) {
  for (int i = 0; i < times; i++) {
    printf("Hello!\n");
  }
}

int main() {
  // Declare a function pointer
  void (*func_ptr)(int);

  // Assign the function's address to the pointer
  func_ptr = greet;

  // Call the function using the pointer
  func_ptr(3); // Outputs "Hello!" three times

  return 0;
}
```

©2025 Beningo Embedded Group, LLC. All Rights Reserved.

Audience POLL Question

How comfortable are you with pointers and function pointers?

a) New to me
b) Beginner
c) Dangerous
d) Expert

©2025 Beningo Embedded Group, LLC. All Rights Reserved.

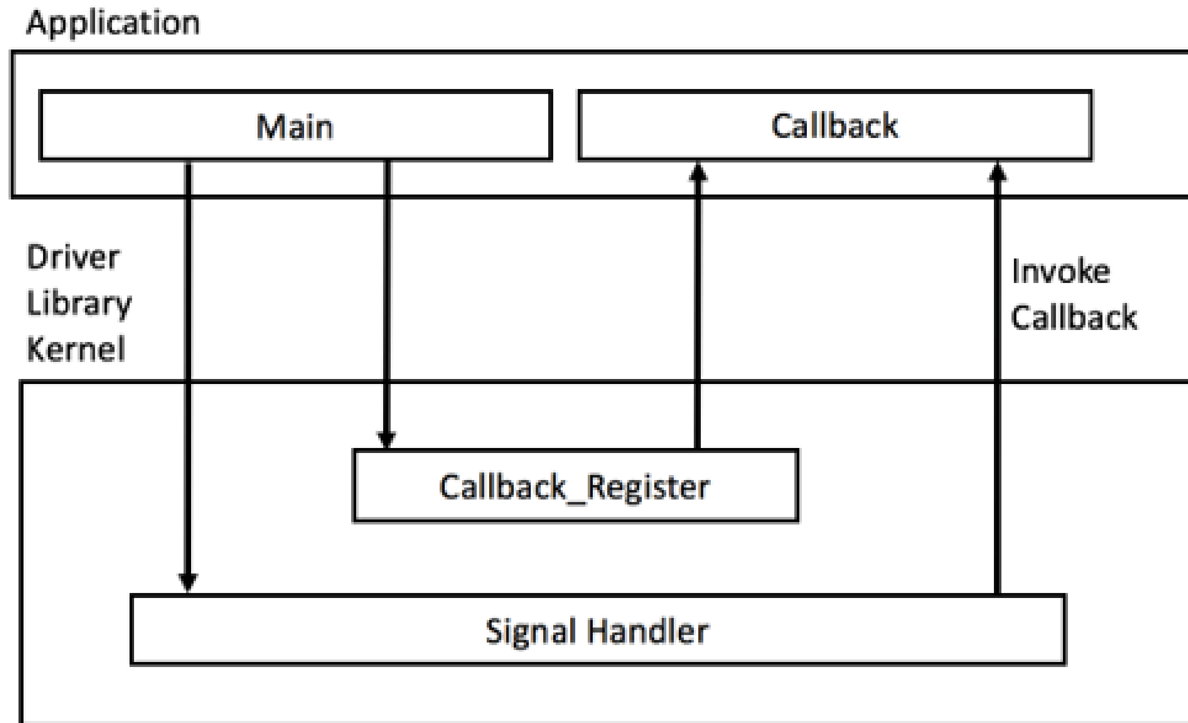
03

Function Pointer Use Cases

©2025 Beningo Embedded Group, LLC. All Rights Reserved.

Function Pointer Use Cases

Callback Functions



https://www.beningo.com/using-callbacks-with-interrupts/

©2025 Beningo Embedded Group, LLC. All Rights Reserved.

Function Pointer Use Cases

Command Processing

```
typedef struct
{
    SystemCommand_t Command;                                /**< Stores the command */
    void (*function)(uint8_t const * const Data);      /**< Function pointer to execute on command */
}CommandRxList_t;

/**
 * Defines an array of all the supported commands and the function that
should
 * be executed when the command is received.
 */
static const CommandRxList_t CommandsList[] =
{
  {CMD_NULL,              Command_Null              },
  {CMD_ARM,               Command_Arm               },
  {CMD_STANDBY,           Command_Standby           },
  {CMD_UPDATE_PARAMS      Command_UpdateParams                      },
  {CMD_CLEAR_FAULTS,      Command_ClearFaults       },
  {CMD_END_LIST,          0x00                      },
};
```

Typedef enum

Function Pointers

©2025 Beningo Embedded Group, LLC. All Rights Reserved.

Function Pointer Use Cases

Schedulers

```
typedef struct
{
  uint16_t Interval;                              /**< Defines how often a task will run  */
  uint32_t LastTick;                      /**< Stores the last tick task was ran  */
  void (*Func)(uint32_t SystemTimeNow);   /**< Function pointer to the task  */
}Task_t;
```

```
/**
 * Task configuration table.  Holds the task
 * interval, last time executed, and the function
 * to be executed.  A continuous task is defined
 * as a task with an interval of 0.  Last time
 * executed is set to 0.
 */
static Task_t Tasks[] =
{
{ INTERVAL_1MS  ,  0, Task_1ms    },
{ INTERVAL_10MS ,  0, Task_10ms   },
{ INTERVAL_40MS ,  0, Task_40ms   },
{ INTERVAL_100MS,  0, Task_100ms  },
{ INTERVAL_200MS,  0, Task_200ms  },
};
```

https://www.beningo.com/152-task-scheduling-with-function-pointers/

©2025 Beningo Embedded Group, LLC. All Rights Reserved.

Function Pointer Use Cases

State Machines

```
/**
 * Call this function to run the state machine
 */
void Sm_Run(void)
{
    // Check to make sure that the state being entered is valid
   if(SmState < NUM_STATES)
   {
       // Dereference the function pointer to run the state
       (*StateMachine[SmState].func)()
   }
};
```

https://www.beningo.com/158-state-machines-with-function-pointers/

©2025 Beningo Embedded Group, LLC. All Rights Reserved.

Audience POLL Question

How much experience do you have with Rust?

a) None
b) beginner
c) intermediate
d) Expert

©2025 Beningo Embedded Group, LLC. All Rights Reserved.

Next Steps

04

©2025 Beningo Embedded Group, LLC. All Rights Reserved.

Going Further

https://beningo.mykajabi.com/

Download Function Pointer Example Code:

- Additional Video on Baremetal Scheduling
- Scheduling Lab Instructions
- Scheduler Source Code

©2025 Beningo Embedded Group, LLC. All Rights Reserved.

Additional Resources

Please consider the resources below:

- Jacob's Blogs
- Jacob's CEC courses
- Embedded Software Academy

- Embedded Bytes Newsletter
  - http://bit.ly/1BAHYXm

www.beningo.com

Consulting | Coaching | Training

©2025 Beningo Embedded Group, LLC. All Rights Reserved.

Next Steps

- Mastering Function Pointers for Baremetal Systems
- Building Cooperative Schedulers and Command Parsers
- Baremetal Performance Analysis and Architecture Design
- Assertions and Validation Techniques for Baremetal
- Managing Interrupts in Baremetal Systems

©2025 Beningo Embedded Group, LLC. All Rights Reserved.

DesignNews

Thank You

Sponsored by

DigiKey