Zephyr® Project Developer Summit

Zephyr® Project
Developer Summit

Pigweed Tokenizer

Al Semjonovs, *Google*

Email: asemjonovs@google.com
Discord: asemjonovs#8232
Github: asemjonovs

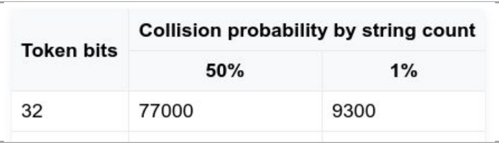
#EMBEDDEDOSSUMMIT

Agenda

- What is Pigweed’s tokenizer?
- Zephyr’s Dictionary Logging Comparison
- How does tokenizing help us?
- How to integrate Pigweed’s tokenization into your project
- Future improvements with tokenization

What is Pigweed's tokenizer?

- Not related to string parsing
- Replaces whole string literal with a 32-bit hash token.
- Why?
  - Reduce binary size by removing string literals from binaries
  - Reduce I/O traffic, RAM, and flash usage.
  - Reduce CPU usage by replacing snprintf calls with simple tokenization code.

	Zephyr Dictionary Logging	**Pigweed Tokenizer**
String mapping	Maps to string address	Maps to 32-bit hash generated by string literal
Probability of Collisions	Strings have a 1:1 mapping to address. Guaranteed no collisions.	
Database Format	JSON dictionary	CSV, Binary, Directory based
Database Portability	Addresses are not guaranteed between builds. Only works with build it was compiled against.	The token hash will be the same between builds and boards using same source. Able to merge tokens from multiple boards and versions to a single database.

Token bits	**Collision probability by string count**	
	50%	**1%**
32	77000	9300

Token Database

- Stores mapping of hash tokens to strings they represent
- Generated from ELF file
- Database Format Types
    - CSV
    - Binary
    - Directory Based
- Update an existing database
    - Captures removal date of unused tokens
- Integrates with CMake and GN builds

Detokenizing

- Language support
  - Python
  - C/C++
  - Typescript
- Web Console
- System Console

  -
    ```
    pigweed$ python3 -m pw_system.console --device
    /dev/ttyUSB0 --token-databases database.csv
    ```

Tokenized Logging Example

Before: Plain text logging

Location	Logging Content	Num bytes
Source contains	`LOG("Battery state: %s; battery voltage: %d mV", state, voltage);`	
Binary contains	`"Battery state: %s; battery voltage: %d mV"`	41
Device Transmits	`"Battery state: CHARGING; battery voltage: 3989 mV"`	49
When Viewed	`"Battery state: CHARGING; battery voltage: 3989 mV"`	

Tokenized Logging Example

After: Tokenized logging

Location	Logging Content	Tokenized Num bytes	Plain Text Num bytes
Source contains	`LOG("Battery state: %s; battery voltage: %d mV", state, voltage);`		

Binary contains	**`d9 28 47 8e`** `(0x8e4728d9)`
	 `(log statement is called with "CHARGING" and 3989 as arguments)`
		4	41

Device Transmits	**`d9 28 47 8e`** `→ Token (4 bytes)`
	 `43 48 41 52 47 49 4E 47 00` `→ "CHARGING" argument (9 bytes)`
	 `0f 95` `→ 3989, as varint (2 bytes)`
		15	49

When Viewed	`"Battery state: CHARGING; battery voltage: 3989 mV"`		

How is tokenizing helping us?

- We're applying tokenization to our logging module.
- Within our Embedded Controller, we have seen a reduction in 14KBs of flash memory, a 6% reduction in our image size.
- Debug logging can be more verbose with tokenization.

Database Management

- How do you want to use your token database?
  - 1:1
  - 1 to many

- Single database supporting multiple EC boards and versions of firmware
  - Chromebooks 8 year support cycle.
- Database size
  - Per EC board: 40 KBs
  - Multiple EC board support: 66 KBs
  - Approximately 1 KB growth per board

How to integrate Pigweed tokenization into your Zephyr project

- It's simple!
- Include
  - `pigweed/Kconfig.zephyr`
- Enable
  - `CONFIG_STD_CPP20=y`
  - `CONFIG_ZEPHYR_PIGWEED_MODULE=y`
  - `CONFIG_PIGWEED_LOG_TOKENIZE=y`
- Update CMake
  - Add database creation dependency

Future improvements with tokenization in EC

- Tokenized strings as %s arguments
  - State names
  - Enumerations
- Tokenized RPC logging

Zephyr® Project
Developer Summit

Thanks!