```
 1: // $Id: uint8print.cpp,v 1.6 2022-01-07 13:20:55-08 - - $
 2:
 3: // Show how to print a uint8_t as a number.
 4: // Options:
 5: // -c as characters
 6: // -n as numbers
 7:
 8: #include <iostream>
 9: #include <memory>
10: #include <string>
11: #include <unordered_map>
12: using namespace std;
13:
14: #ifdef __GNUG__
15:
16:    #include <cxxabi.h>
17:    string demangle (const char* name) {
18:       int status = 0;
19:       unique_ptr<char,void(*)(void*)> result {
20:          abi::__cxa_demangle (name, nullptr, nullptr, &status),
21:          std::free,
22:       };
23:       return status == 0 ? result.get() : name;
24:    }
25:
26: #else
27:
28:    string demangle (const char* name) {
29:       return name;
30:    }
31:
32: #endif
33:
34: string basename (const string &name) {
35:    return name.substr (name.rfind ('/') + 1);
36: }
37:
38: template <typename T>
39: void print() {
40:    const char* type = typeid(T).name();
41:    cout << "Printing as " << type << " = " << demangle (type) << endl;
42:    cout << "[";
43:    for (uint8_t n = 0; n <= 9; ++n) cout << T(n);
44:    cout << "]" << endl;
45: }
46:
47: unordered_map<string,void(*)()> fns {
48:    {"-8", print<uint8_t>},
49:    {"-i", print<int>},
50:    {"-u", print<unsigned>},
51: };
52:
53: int main (int argc, char** argv) {
54:    string execname = basename (argv[0]);
55:    string opt = argc == 1 ? "" : argv[1];
56:    auto fn = fns.find (opt);
57:    if (argc != 2 or fn == fns.end()) {
58:       cerr << "Usage: " << execname << " -8|-i|-u" << endl;
```

```
59:       return EXIT_FAILURE;
60:    }
61:    fn->second();
62:    return EXIT_SUCCESS;
63: }
64:
65: //TEST// for cmd in "uint8print -8" \
66: //TEST//             "uint8print -8 | cat -t" \
67: //TEST//             "uint8print -8 | od -c" \
68: //TEST//             "uint8print -i" \
69: //TEST//             "uint8print -u"
70: //TEST// do
71: //TEST//    echo ""
72: //TEST//    echo bash: $cmd
73: //TEST//    sh -c "$cmd"
74: //TEST// done >uint8print.out
75: //TEST// mkpspdf uint8print.ps uint8print.cpp uint8print.out
76:
```

```
 1:
 2: bash: uint8print -8
 3: Printing as h = unsigned char
 4: [\000\001\002\003\004\005\006\007      ]
 5:
 6: bash: uint8print -8 | cat -t
 7: Printing as h = unsigned char
 8: [^@^A^B^C^D^E^F^G^H^I]
 9:
10: bash: uint8print -8 | od -c
11: 0000000   P   r   i   n   t   i   n   g       a   s       h       =
12: 0000020   u   n   s   i   g   n   e   d       c   h   a   r  \n   [  \0
13: 0000040 001 002 003 004 005 006  \a  \b  \t   ]  \n
14: 0000053
15:
16: bash: uint8print -i
17: Printing as i = int
18: [0123456789]
19:
20: bash: uint8print -u
21: Printing as j = unsigned int
22: [0123456789]
```