POSIX Nexus serves as a comprehensive cross-language reference hub that explores the implementation and behavior of POSIX-compliant functionality across a diverse set of programming environments. Built atop the foundational IEEE Portable Operating System Interface (POSIX) standards, this project emphasizes compatibility, portability, and interoperability between operating systems.

**Abstract**

## Contents

# I Specifications

## I Grammer

```
       <symbol>    <prefix-token> <name> <order> <terminator>
 <prefix-token>    ”NX” | ”nX” | ”nx” | ”Nx”
        <order>    <order-usa> :in: ”NX”
                 | <order-aus> :in: ”nX”
                 | <order-asu> :in: ”nx”
                 | <order-sua> :in: ”Nx”
         <name>    <letter>
                 | <letter> <name>
       <letter>    ”a” | ”b” | ”c” | ”d” | ”e” | ”f” | ”g” | ”h” | ”i” | ”j”
                 | ”k” | ”l” | ”m” | ”n” | ”o” | ”p” | ”q” | ”r” | ”s” | ”t”
                 | ”u” | ”v” | ”w” | ”x” | ”y” | ”z”
                 | ”A” | ”B” | ”C” | ”D” | ”E” | ”F” | ”G” | ”H” | ”I” | ”J”
                 | ”K” | ”L” | ”M” | ”N” | ”O” | ”P” | ”Q” | ”R” | ”S” | ”T”
                 | ”U” | ”V” | ”W” | ”X” | ”Y” | ”Z”
    <order-usa>    <unsigned> | <signed> | <auto>
    <order-aus>    <auto> | <unsigned> | <signed>
    <order-asu>    <auto> | <signed> | <unsigned>
    <order-sua>    <signed> | <unsigned> | <auto>
          <int>    v | V : short
                 | i | I : int
                 | l | L : long
        <float>    f | F : float
                 | e | E : double
         <auto>    <size>
       <signed>    <float>
                 | <int>
                 | <auto>
     <unsigned>    <int>
                 | <auto>
         <size>    b | B : 1 byte
                 | w | W : 2 bytes
                 | d | D : 4 bytes
                 | a | A : 8 bytes
                 | o | O : 16 bytes
                 | h | H : 32 bytes
                 | s | S : 64 bytes
                 | p | P : 128 bytes
                 | x | X : 256 bytes
   <terminator>    ”F” | ”E” | ”M” | ”H” | ”S” | ”U” | ”C” | ”G” | ”T”
```

## I Prefix

**Prefix Token Summary**

| Prefix | Ordering Rule |
|---|---|
| NX | unsigned > signed > auto |
| nX | auto > unsigned > signed |
| nx | auto > signed > unsigned |
| Nx | signed > unsigned > auto |

**Prefix Token Conductor — The Permutation Glyph**

- **Purpose** Define the permutation ordering of type-groups (U, S, A) for the identifier
- **Tokens**
  - **"NX"** U > S > A
  - **"nX"** A > U > S
  - **"nx"** A > S > U
  - **"Nx"** S > U > A
- **Invariant** Auto (A) is never allowed in the middle position
- **Use Case** Determines which type-group is selected first when resolving the identifier's type signature

## I Grouping

### Order Group Summary

| Order Group | Resolution Priority |
|---|---|
| `<order-usa>` | unsigned > signed > auto |
| `<order-aus>` | auto > unsigned > signed |
| `<order-asu>` | auto > signed > unsigned |
| `<order-sua>` | signed > unsigned > auto |

### Order Group Conductor — The Resolution Glyph

- **Purpose** Define the priority sequence for resolving type-groups (unsigned, signed, auto)
- **Groups**
  - **<order-usa>** unsigned > signed > auto
  - **<order-aus>** auto > unsigned > signed
  - **<order-asu>** auto > signed > unsigned
  - **<order-sua>** signed > unsigned > auto
- **Invariant** Auto is never permitted in the middle position; only four permutations are valid
- **Use Case** Selected by prefix-token to determine which type-group is attempted first during type resolution

## Type Group Summary

| Group | Description |
|---|---|
| `<signed>` | Accepts floats, ints, or auto-sized types |
| `<unsigned>` | Accepts ints or auto-sized types |
| `<auto>` | Resolves to a size token (b, w, d, a, o, h, s, p, x) |
| `<int>` | v/V = short, i/I = int, l/L = long |
| `<float>` | f/F = float, e/E = double |

## Type Group Conductor — The Resolution Families

- **Purpose** Define the semantic families of types that can be selected by the prefix ordering
- **Groups**
  - **<signed>** Accepts *float*, *int*, or *auto* types
  - **<unsigned>** Accepts *int* or *auto* types
  - **<auto>** Resolves to a physical width token (b–x)
  - **<int>** v/V = short; i/I = int; l/L = long
  - **<float>** f/F = float; e/E = double
- **Invariant** Signed types include floats; unsigned types do not; auto defers to size resolution
- **Use Case** These groups form the selectable branches inside each <order-*> permutation

**Size Token Summary**

| Token | Width |
|---|---|
| b / B | 1 byte |
| w / W | 2 bytes |
| d / D | 4 bytes |
| a / A | 8 bytes |
| o / O | 16 bytes |
| h / H | 32 bytes |
| s / S | 64 bytes |
| p / P | 128 bytes |
| x / X | 256 bytes |

**Size Token Conductor — The Physical Width Ladder**

- **Purpose** ⇝ Define the primitive physical widths used by <auto> and symbolic-width constructions
- **Width** ⇝
  - **b/B** ⇝ 1 byte
  - **w/W** ⇝ 2 bytes
  - **d/D** ⇝ 4 bytes
  - **a/A** ⇝ 8 bytes
  - **o/O** ⇝ 16 bytes
  - **h/H** ⇝ 32 bytes
  - **s/S** ⇝ 64 bytes
  - **p/P** ⇝ 128 bytes
  - **x/X** ⇝ 256 bytes
- **Invariant** ⇝ This ladder represents the complete set of primitive machine widths; symbolic widths must be composed from these atoms
- **Use Case** ⇝ Used by <auto> to resolve size, and by symbolic-width forms such as x08x to construct larger composite widths

**Terminator Summary**

| Terminator | Meaning |
|---|---|
| F | Function |
| E | Enum |
| S | Struct |
| H | Header guard |
| U | Union |
| G | Guard (used in .c files; .h uses H) |
| C | Constant |
| M | Macro |
| T | Typedef (fallback category) |

**Terminator Summary**

| Terminator | Meaning |
|---|---|
| F | Function |
| E | Enum |
| S | Struct |
| H | Header guard |
| U | Union |
| G | Guard (used in .c files; .h uses H) |
| C | Constant |
| M | Macro |
| T | Typedef (fallback category) |

## I Rationale

rat

## I Semantics

sem

## I Virtual Machine

vm