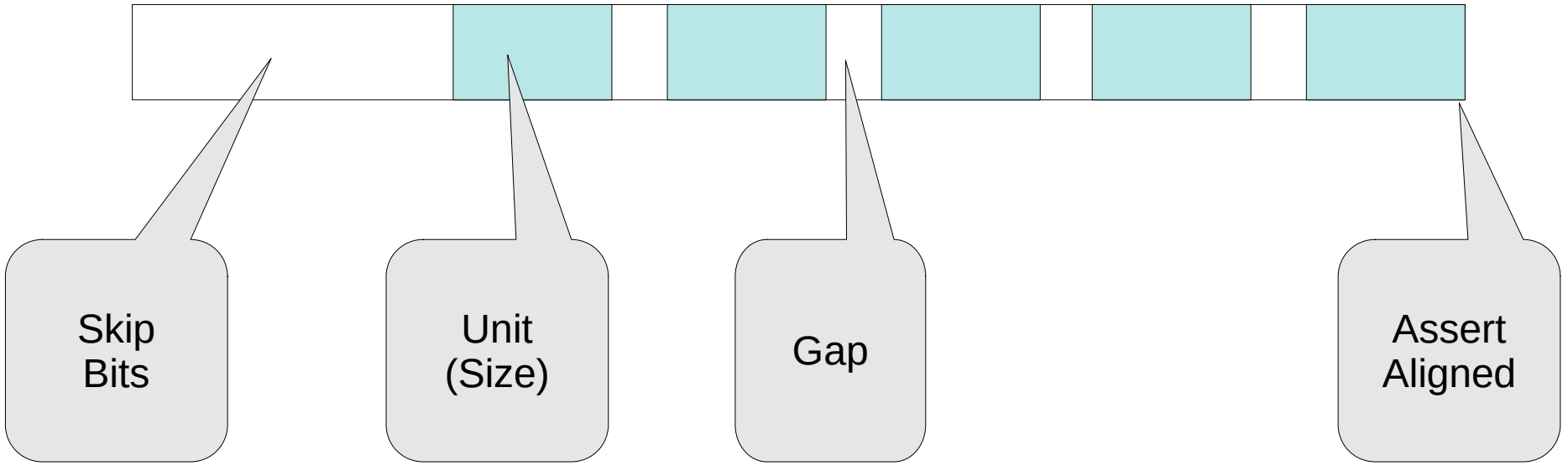
bdd

(http://bdd.sourceforge.net)

Bit Stream Manipulation Command
For Unix/Linux/POSIX Command Line
Implemented in Python
Very Flexible – Not Very Efficient
Freeware

By
Esa Turtiainen

A Bit Stream

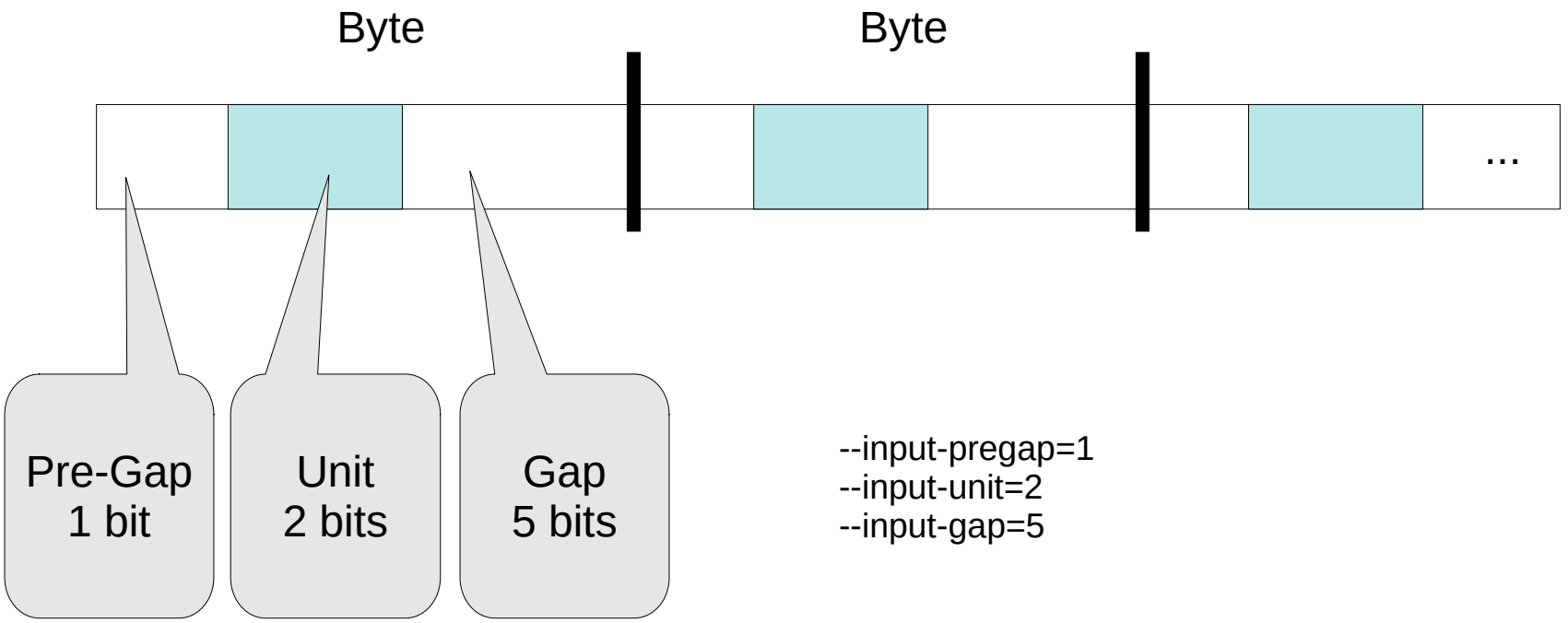


All measures in bits, not bytes

e.g.
--input-skip-bits=128
--input-unit=7
--input-gap=3
--input-assert-aligned

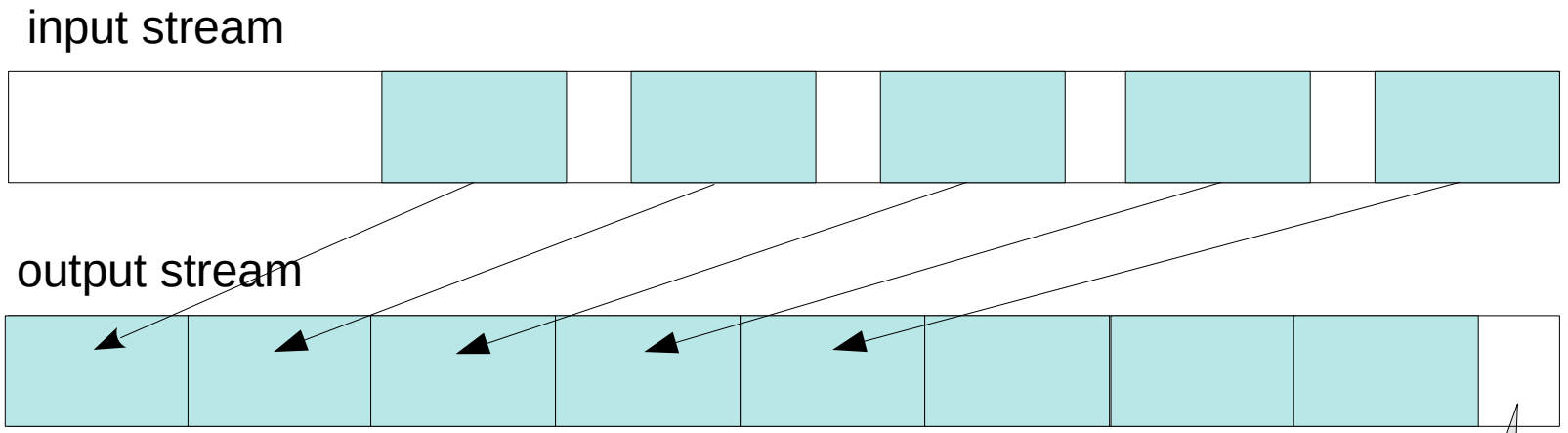
Gives an error message if not aligned to a byte

Example, Bits from a Byte Stream



--input-pregap is more convenient but identical to:
  --input-skip-bits <pregap more>
  --input-gap <pregap more>

Output Bit Stream



If the size of the output unit is bigger than the input unit:
  Left bits are truncated
If the size of the output unit is smaller than the input unit:
  Left bits are filled with zeros

Unit is handled as if it was an integer

Example, Rearrange Bit Streams

- Take most significant bit of a byte stream
  - ➔ `bdd --input-unit=1 --input-gap=7 \`
    `--output-unit=8`
    - examples assume that the input is piped to the bdd like “cat file | bdd”, you can also use --input-file
    - 1 bit will be filled to 8 bits: most significant bits = 0
    - Output byte stream of values 0x00 and 0x01
- Pack least significant bit of bytes to a bit stream
  - ➔ `bdd --input-pregap=7 –input-unit=1 \`
    `--output-unit=1`
    - If input is 9 bytes all bits ones, the output will be two bytes 0xFF and 0x80

“Fake” Bit Streams

- Input:
  - Zeros
  - Ones
  - Random
    - /dev/urandom
  - Counter
    - Starts from zero
    - Every unit previous plus one
- Output:
  - Print unit as integer
    - one integer at line
    - meant for programs
  - Print unit as hex
    - human-readable lines
  - Print unit as bits
    - human-readable lines

Default Unit Size

- Rule to remember:

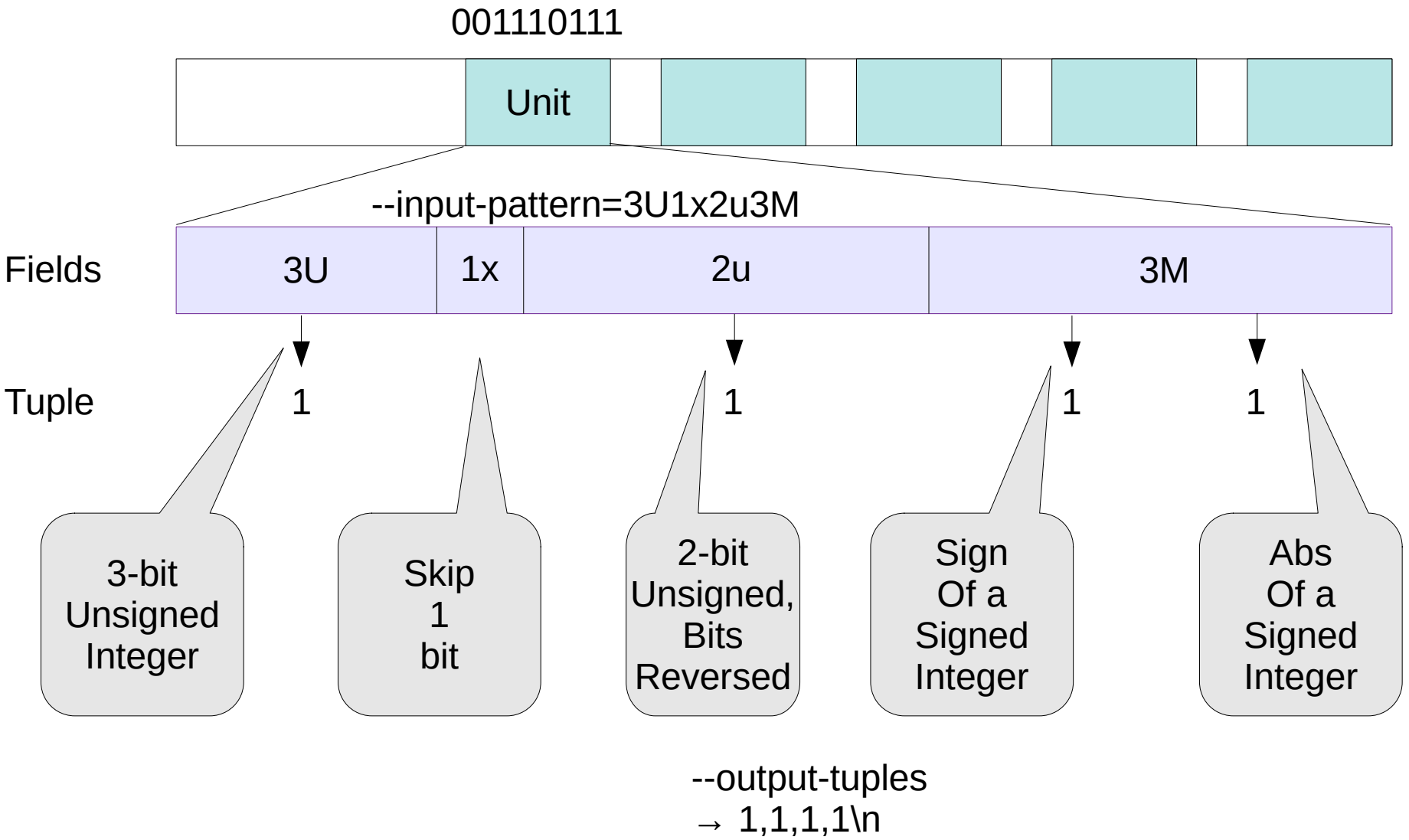
  ```
  bdd <input >output
  ```

  just copies 8-bit bytes from input to output
- Default input unit is 8 bits
- Default output unit is 8 bits
  (will likely change in the future:
  the same as the input unit)

Tuple

- Tuple is a unit that is split into fields
- It is possible to give fields types so that e.g. some bits are interpreted as signed integers or floating point numbers

Tuple Unpack



Pattern Characters (Input)

- U – Unsigned integer
- x – Skip bits
- M – Signed integer as two unsigned integers

- U can be used to unpack any bit pattern and it will pack identically
- M assumes that the field is 2's complement integer
- M will return two values: first the sign (0=positive, 1=negative) and then abs(value)

“Fake” Tuple Streams

- --input-tuples reads human-readable tuples directly from the input stream
- --output-tuples writes human-readable tuples directly to the output stream
- Comma separated fields, one line per tuple:
  - `1,1.2,″abba″`

Example, Look inside a Bit Stream

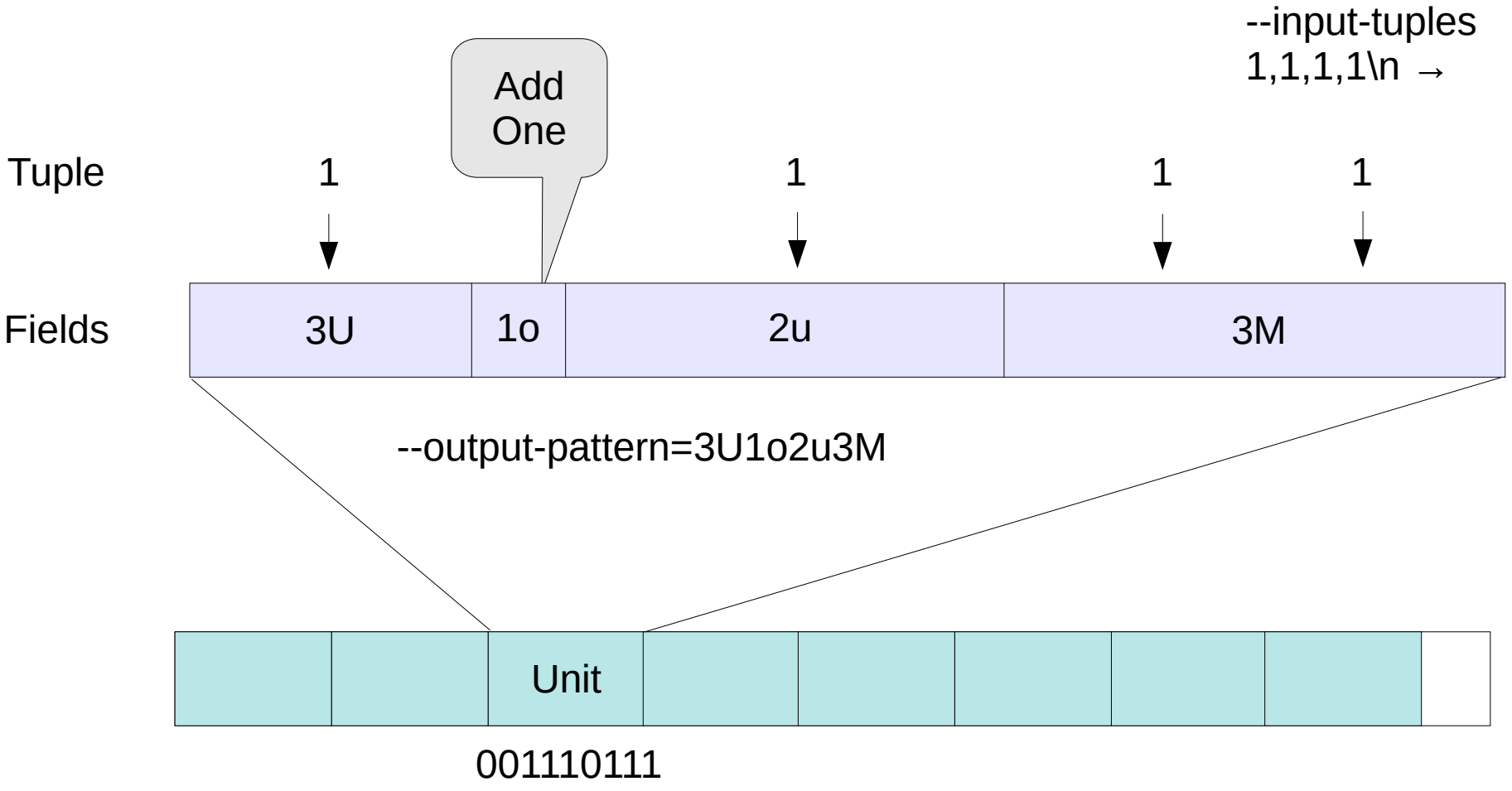
- Interpret units to human-readable lines:
  - `bdd --input-pattern=3U1x2u3M --output-tuples`
  - Prints result as comma separated list, one unit at a line, e.g.

```
1,1,1,1
3,2,0,2
...
```

Typed Patter Characters

- The following pattern characters do not return integer value. They print nicely with --output-tuples, but if you try to repack or manipulate them, pattern must exactly match the type.
- S – signed integer
- F – 32-bit IEEE floating point number
- D – 64-bit IEEE floating point number
- C – characters (printed as UTF-8)

Tuple Pack


--input-tuples
1,1,1,1\n →
Add
One
Tuple
1
1
1
1
Fields
3U
1o
2u
3M
--output-pattern=3U1o2u3M
Unit
001110111

Pattern Characters (Output)

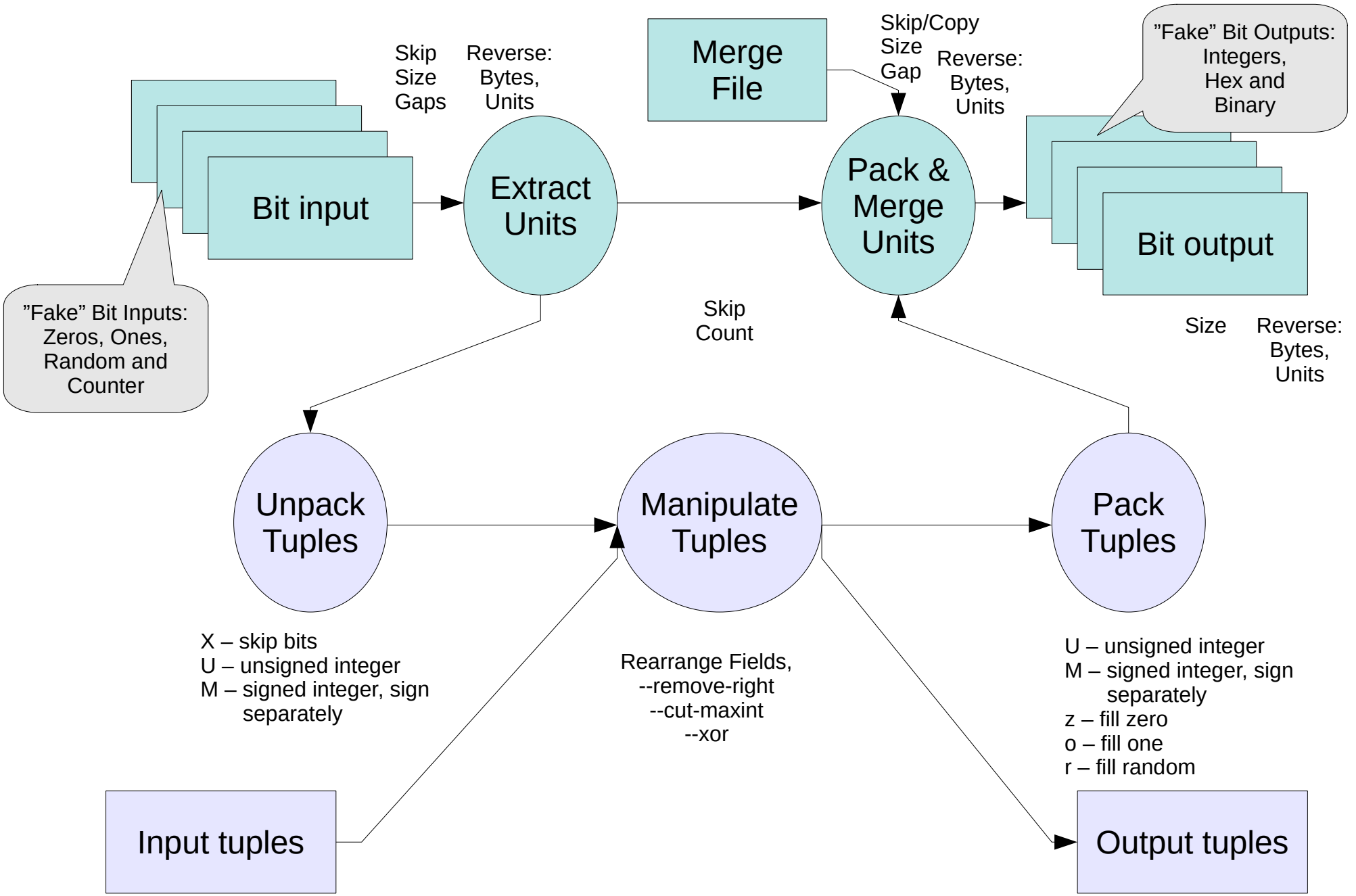
- Output patterns have some special characters:
- z – Fill bits with zeros
- o – Fill bits with ones
- r – Fill bits with random bits (/dev/urandom)

Examples, Pack from Text

- Build a hex number from integers:
  - ```
    echo -en ”1,2\n3,4” | bdd --input-tuples \
        --output-pattern=4U4U | od -t x1
    ```
    - Prints (hex) 12 34
- Build a stream of floating point numbers:
  - ```
    echo -en ”1.0\n2.0\n” | \
    bdd --input-tuples --output-pattern=64D
    ```

Example, Combined

- ```
  echo 1,1,1,1 | \
  bdd --input-tuples --output-pattern=3U1o2u3M | \
  bdd --input-pattern=3U1x2u3M --output-tuples
  ```
- Prints:
  - `1,1,1,1`
  - `0,0,0,0`
- 9-bit output pattern outputs two bytes and when read, the second is read as all-zeros
  ```


Bit input
Skip
Size
Gaps
Reverse:
Bytes,
Units
Extract
Units
Merge
File
Skip/Copy
Size
Gap
Reverse:
Bytes,
Units
Pack &
Merge
Units
"Fake" Bit Outputs:
Integers,
Hex and
Binary
Bit output
"Fake" Bit Inputs:
Zeros, Ones,
Random and
Counter
Skip
Count
Size
Reverse:
Bytes,
Units
Unpack
Tuples
Manipulate
Tuples
Pack
Tuples
X – skip bits
U – unsigned integer
M – signed integer, sign
separately
Rearrange Fields,
--remove-right
--cut-maxint
--xor
U – unsigned integer
M – signed integer, sign
separately
z – fill zero
o – fill one
r – fill random
Input tuples
Output tuples

Example, Manipulating Tuples

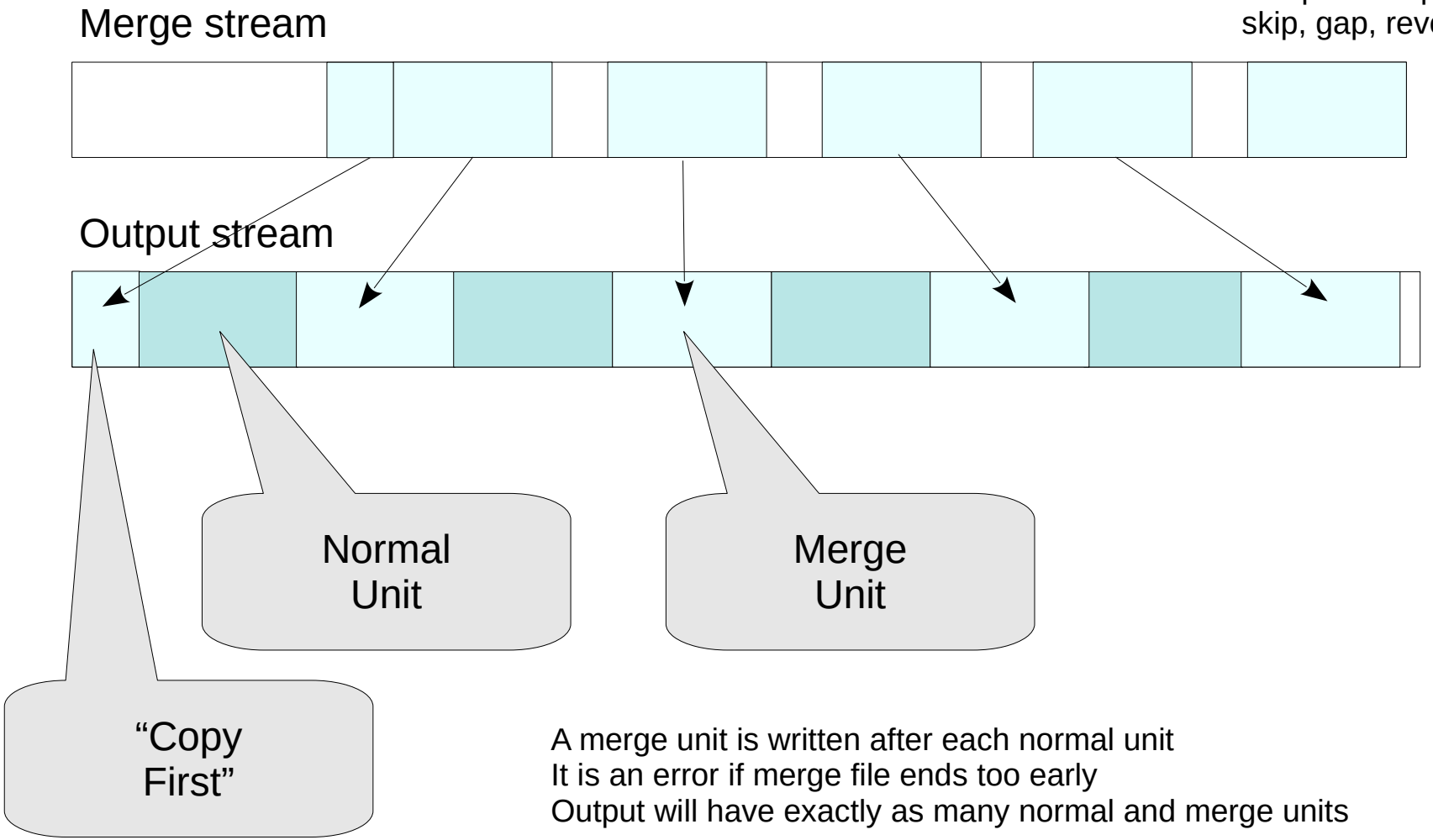
- Input of 1+2 bits in stream, turn bits 2+1
  - ➔ `bdd --input-pattern=1U2U \`
    `--rearrange=1,0 \`
    `--output-pattern=2U1U`
    - Fields are numbered starting from 0
- Input is signed bytes, we want the most significant bit of the magnitude, after -128 is truncated to -127:
  - ➔ `bdd --input-pattern=8M \`
    `--rearrange=-1 --cut-maxint=0,127 \`
    `--remove-right=0,6 --output-unit=1`
    - -1 is the last field in tuple, -2 second last etc.
- Options are executed in the specified order

Reversing Bits

- Reversing bits can be quite a mess...
- bdd implements
  - Reversing bits of every byte read
  - Reversing bits of every unit read
  - Matching patterns in reverse bit order
  - Reversing bits of every unit written
  - Reversing bits of every byte written
- --in/output-little-endian is a synonym for combined reversing of both bytes and unit

Merge Bit Stream

Merge file options
are mostly identical
to input file options:
skip, gap, reverse



A merge unit is written after each normal unit
It is an error if merge file ends too early
Output will have exactly as many normal and merge units

Example, Merge

- Print first 16 bytes of /etc/passwd file as hex preceded by a three-digit hex address of the byte:

```
➔ bdd --input-counter --count=16 \
    --input-unit=12 --output-unit=12 \
    --merge-file=/etc/passwd \
    --output-hex

000 72 001 6f 002 6f 003 74 004 3a 005 78 006 3a 007 30
008 3a 009 30 00a 3a 00b 72 00c 6f 00d 6f 00e 74 00f 3a
```