## NAME

TAP::Parser::Multiplexer - Multiplex multiple TAP::Parsers

## VERSION

Version 3.30

## SYNOPSIS

```
use TAP::Parser::Multiplexer;

my $mux = TAP::Parser::Multiplexer->new;
$mux->add( $parser1, $stash1 );
$mux->add( $parser2, $stash2 );
while ( my ( $parser, $stash, $result ) = $mux->next ) {
    # do stuff
}
```

## DESCRIPTION

`TAP::Parser::Multiplexer` gathers input from multiple TAP::Parsers. Internally it calls select on the input file handles for those parsers to wait for one or more of them to have input available.

See *TAP::Harness* for an example of its use.

## METHODS

### Class Methods

#### new

```
my $mux = TAP::Parser::Multiplexer->new;
```

Returns a new `TAP::Parser::Multiplexer` object.

### Instance Methods

#### add

```
$mux->add( $parser, $stash );
```

Add a TAP::Parser to the multiplexer. `$stash` is an optional opaque reference that will be returned from `next` along with the parser and the next result.

#### parsers

```
my $count   = $mux->parsers;
```

Returns the number of parsers. Parsers are removed from the multiplexer when their input is exhausted.

#### next

Return a result from the next available parser. Returns a list containing the parser from which the result came, the stash that corresponds with that parser and the result.

```
my ( $parser, $stash, $result ) = $mux->next;
```

If `$result` is undefined the corresponding parser has reached the end of its input (and will automatically be removed from the multiplexer).

When all parsers are exhausted an empty list will be returned.

```
if ( my ( $parser, $stash, $result ) = $mux->next ) {
```

```
            if ( ! defined $result ) {
                # End of this parser
            }
            else {
                # Process result
            }
        }
        else {
            # All parsers finished
        }
```

## See Also

*TAP::Parser*

*TAP::Harness*