

julia memo

Masaya Kameyama

2021-07-29

```
versioninfo()
```

```
Julia Version 1.6.1
Commit 6aaedecc44 (2021-04-23 05:59 UTC)
Platform Info:
  OS: macOS (x86_64-apple-darwin18.7.0)
  CPU: Intel(R) Core(TM) i7-8557U CPU @ 1.70GHz
  WORD_SIZE: 64
  LIBM: libopenlibm
  LLVM: libLLVM-11.0.1 (ORCJIT, skylake)
Environment:
  JULIA_NUM_THREADS = 8
```

julia

for

for x

```
x = rand(1:5, 10, 3)
```

10×3 Matrix{Int64}:

```
2  1  3
1  1  5
5  1  1
4  1  2
3  5  5
5  4  3
4  4  4
5  4  2
```

1 5 5
3 5 2

for :
end

```
for e in x  
  println(e)  
end
```

2
1
5
4
3
5
4
5
1
3
1
1
1
1
1
5
4
4
4
5
5
3
5
1
2
5
3
4
2
5
2

eachrow :

```

for row in eachrow(x)
    println(row)
end

```

```

[2, 1, 3]
[1, 1, 5]
[5, 1, 1]
[4, 1, 2]
[3, 5, 5]
[5, 4, 3]
[4, 4, 4]
[5, 4, 2]
[1, 5, 5]
[3, 5, 2]

```

```

for col in eachcol(x)
    println(col)
end

```

```

[2, 1, 5, 4, 3, 5, 4, 5, 1, 3]
[1, 1, 1, 1, 5, 4, 4, 4, 5, 5]
[3, 5, 1, 2, 5, 3, 4, 2, 5, 2]

```

y

```

y=zeros{Int,3} for i=1:10
for i=1:10
    for j=1:3
        y[i][j]=rand(1:5)
    end
end
end
y

```

10-element Vector{Vector{Int64}}:

```

[4, 2, 4]
[4, 2, 1]
[1, 2, 1]
[5, 3, 5]
[3, 5, 3]

```

```
[2, 1, 3]
[3, 5, 4]
[5, 3, 4]
[4, 5, 2]
[1, 3, 1]
```

```
:
```

```
for row in y
    println(row)
end
```

```
[4, 2, 4]
[4, 2, 1]
[1, 2, 1]
[5, 3, 5]
[3, 5, 3]
[2, 1, 3]
[3, 5, 4]
[5, 3, 4]
[4, 5, 2]
[1, 3, 1]
```

```
filter      :
```

```
?filter
```

```
search: filter filter! fieldtype fieldtypes
```

```
filter(f, a)
```

Return a copy of collection **a**, removing elements for which **f** is **false**. The function **f** is passed one argument.

!!! compat “Julia 1.4” Support for **a** as a tuple requires at least Julia 1.4.

Examples

```
julia> a = 1:10
1:10

julia> filter(isodd, a)
5-element Vector{Int64}:
 1
 3
 5
 7
 9
```

`filter(f, d::AbstractDict)`

Return a copy of `d`, removing elements for which `f` is `false`. The function `f` is passed `key=>value` pairs.

Examples

```
julia> d = Dict{1=>"a", 2=>"b"}
Dict{Int64, String} with 2 entries:
 2 => "b"
 1 => "a"

julia> filter(p->isodd(p.first), d)
Dict{Int64, String} with 1 entry:
 1 => "a"
```

`filter(f, itr::SkipMissing{<:AbstractArray})`

Return a vector similar to the array wrapped by the given `SkipMissing` iterator but with all missing elements and those for which `f` returns `false` removed.

!!! compat “Julia 1.2” This method requires Julia 1.2 or later.

Examples

```
julia> x = [1 2; missing 4]
2×2 Matrix{Union{Missing, Int64}}:
 1      2
missing 4
```

```
julia> filter(isodd, skipmissing(x))
1-element Vector{Int64}:
 1
```

```
filter      :
```

```
x = reshape([rand{Int} for i=1:10*3], (:, 3))
```

```
10×3 Matrix{Int64}:
-5225426650602014922 -7897912436381683945 -3211480135654296764
-2367706378189222118 -8074233384091202526  7140575899278393832
-3701204663351764859  8118911886810389263 -2465634172456524632
-2009298108373942027  2477122666332687763 -9127782429389187925
-7849043397713143979 -4501639957282787028 -2523742365264192937
-5756877372530701820 -7147116965065108201 -8647207541707448429
-6152057904350993939 -7050448553908782041  6147233744355870367
 5026289808326491480 -7763311989044274567  3433803153276873203
-7336383049661808522  3788869708303507081 -4376110160490542272
-129002948401320750  4228450907397190871  1524287850472670616
```

```
filter(isodd, skipmissing(x))
```

```
17-element Vector{Int64}:
-3701204663351764859
-2009298108373942027
-7849043397713143979
-6152057904350993939
-7897912436381683945
 8118911886810389263
 2477122666332687763
-7147116965065108201
-7050448553908782041
-7763311989044274567
 3788869708303507081
 4228450907397190871
```

```
-9127782429389187925
-2523742365264192937
-8647207541707448429
6147233744355870367
3433803153276873203
```

```
2 :
```

```
filter(x->iseven(x[2]), x)
```

LoadError: BoundsError

```
filter :
```

```
x[x[:,2] .%2 .==0,:]
```

2×3 Matrix{Int64}:

```
-2367706378189222118 -8074233384091202526 7140575899278393832
-7849043397713143979 -4501639957282787028 -2523742365264192937
```

```
x[iseven.(x[:,2]),:]
```

2×3 Matrix{Int64}:

```
-2367706378189222118 -8074233384091202526 7140575899278393832
-7849043397713143979 -4501639957282787028 -2523742365264192937
```

```
x[findall(a -> iseven(x[a,2]), 1:size(x)[1]),:]
```

2×3 Matrix{Int64}:

```
-2367706378189222118 -8074233384091202526 7140575899278393832
-7849043397713143979 -4501639957282787028 -2523742365264192937
```

```
x[findall(iseven,x[:,2]),:]
```

2×3 Matrix{Int64}:

```
-2367706378189222118 -8074233384091202526 7140575899278393832
-7849043397713143979 -4501639957282787028 -2523742365264192937
```

list join

atcoder for .

```
l=rand(0:9,100);
```

```
@time println(join(l))
```

90034643504554423627381022268592179910026311050060126616203604065552477031625769296998777282
0.001320 seconds (228 allocations: 10.438 KiB)

```
s=""  
for n in l  
    s*=string(n)  
end  
@time println(s)
```

90034643504554423627381022268592179910026311050060126616203604065552477031625769296998777282
0.000147 seconds (21 allocations: 640 bytes)

```
function j(l)  
    println(join(l))  
end  
@time j(l)
```

90034643504554423627381022268592179910026311050060126616203604065552477031625769296998777282
0.004714 seconds (1.05 k allocations: 59.887 KiB, 93.06% compilation time)

```
function jj(l)  
    s=""  
    for n in l  
        s*=string(n)  
    end  
    println(s)  
end  
@time jj(l)
```


90034643504554423627381022268592179910026311050060126616203604065552477031625769296998777282
0.012173 seconds (4.91 k allocations: 249.832 KiB, 96.46% compilation time)

1.

```
@time begin
s=""
for n in 1
    s*=string(n)
end
println(s)
end
```

90034643504554423627381022268592179910026311050060126616203604065552477031625769296998777282
0.000244 seconds (421 allocations: 20.469 KiB)