# CSE 531: Distributed and Multiprocessor Operating Systems

Fall 2017

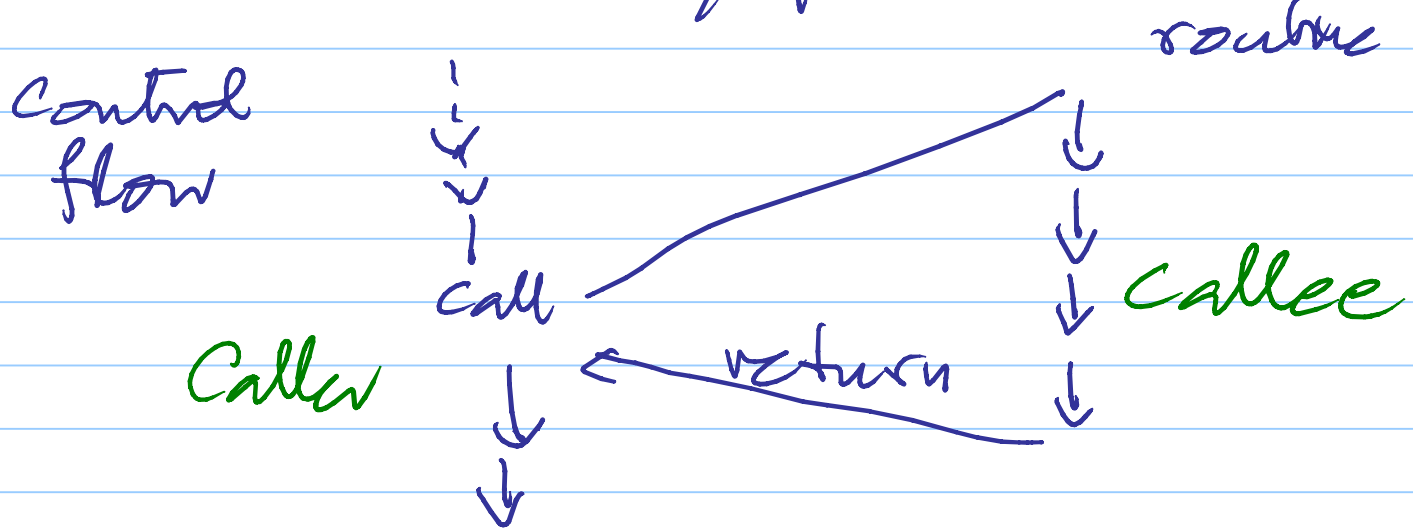
Class: 10 Date: 9/19

Note Title

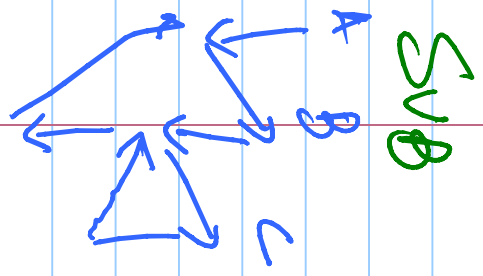
- Semaphores → mutex, sync
- Producer Consumer
- Readers & writers (see code)

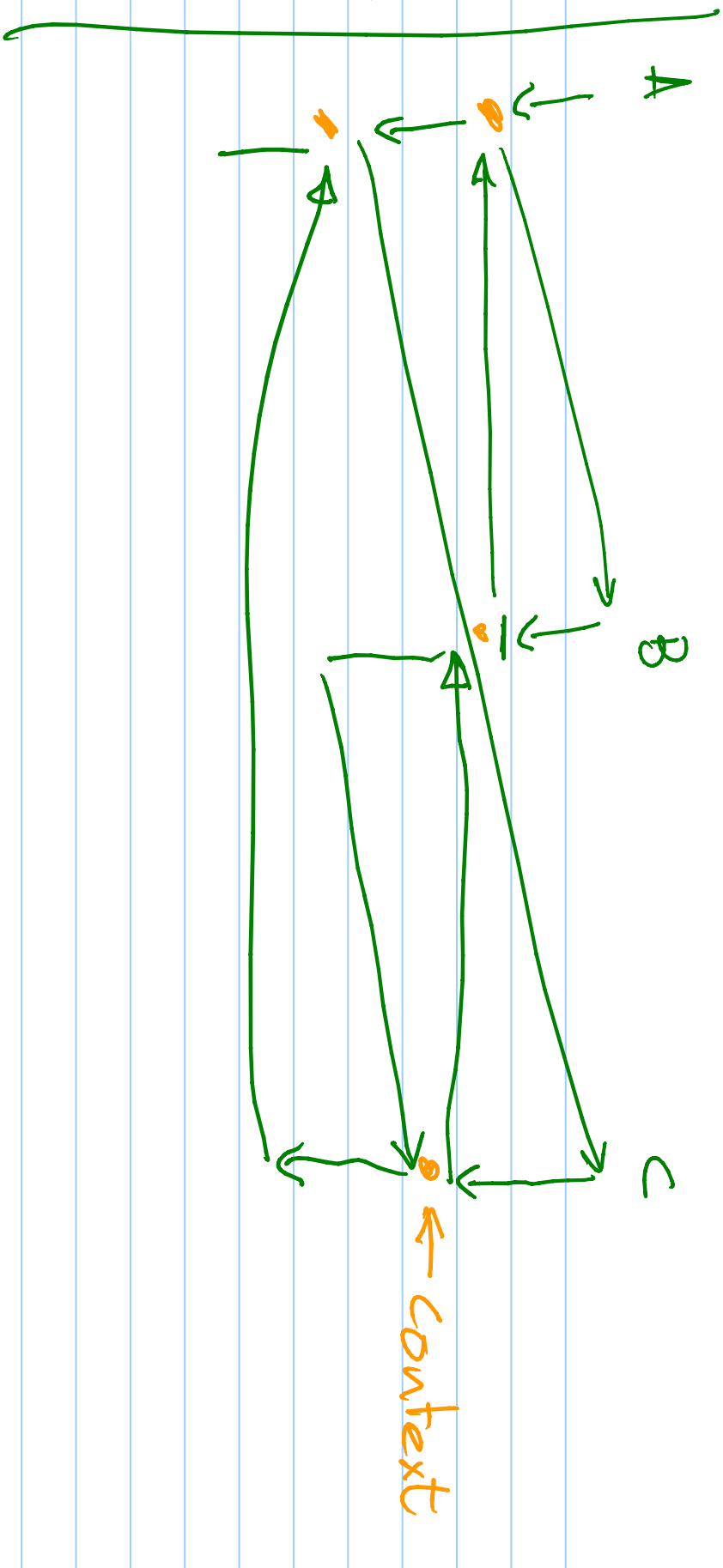
# Coroutines

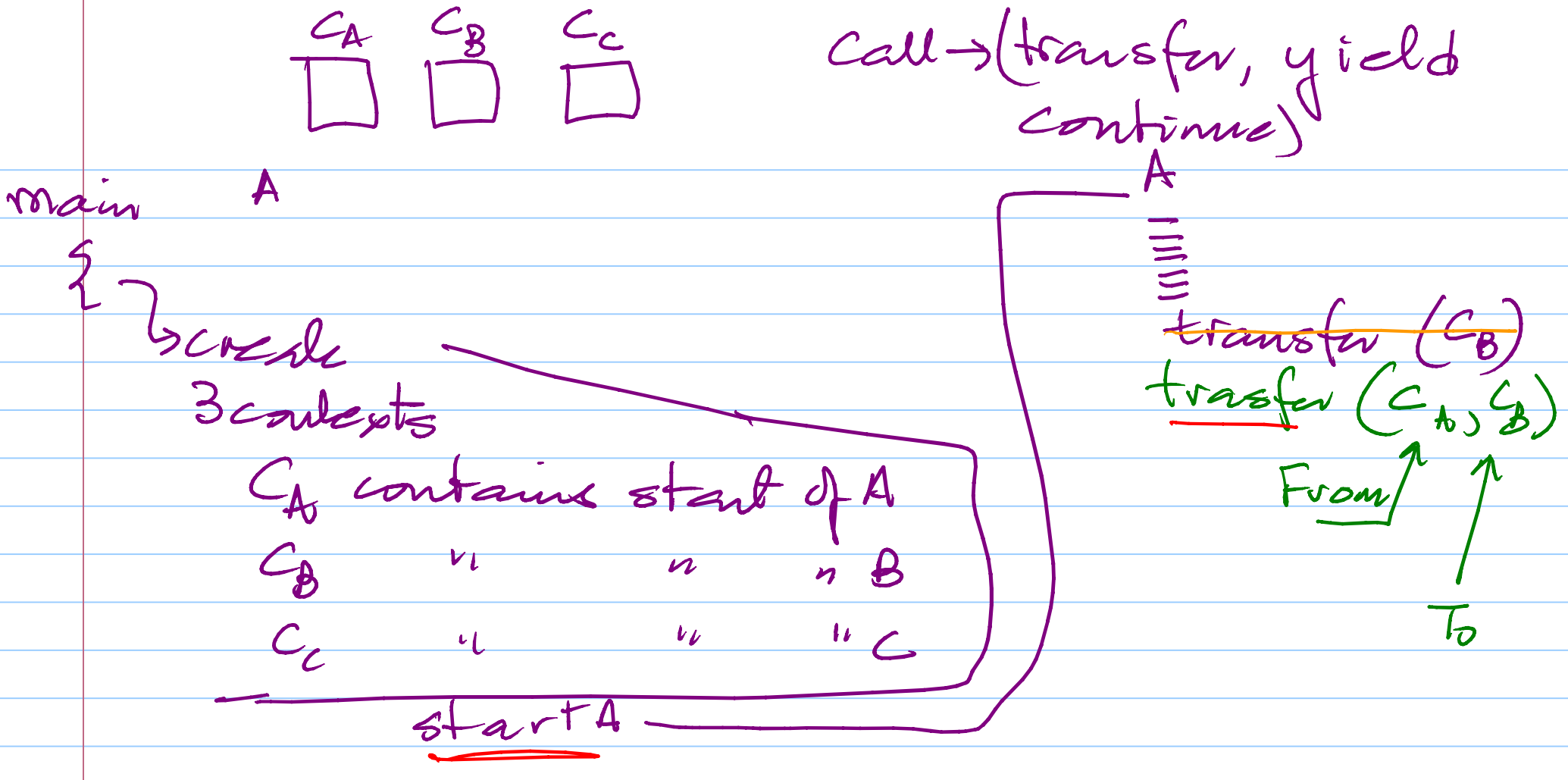
↳ subroutine / function



source A
source B
Top
Bottom


A
B
C
Sub


A
B
C
Context


$C_A$ $C_B$ $C_C$
call → (transfer, yield continue)
A
A
main
{
→ create 3 contexts
$C_A$ contains start of A
$C_B$ " " " B
$C_C$ " " " C
startA
transfer ($C_B$)
transfer ($C_A$, $C_B$)
From
$T_0$

# Coroutines & threads

- special case of coroutines
  - → from → to determined by scheduler
  - → point of switch can be explicit or implicit ← interrupt

# Threads

startthread (f)

f: function

+ stack

↳ allocate context_variable (cv)

put context of f → cv

put cv into a Queue of contexts

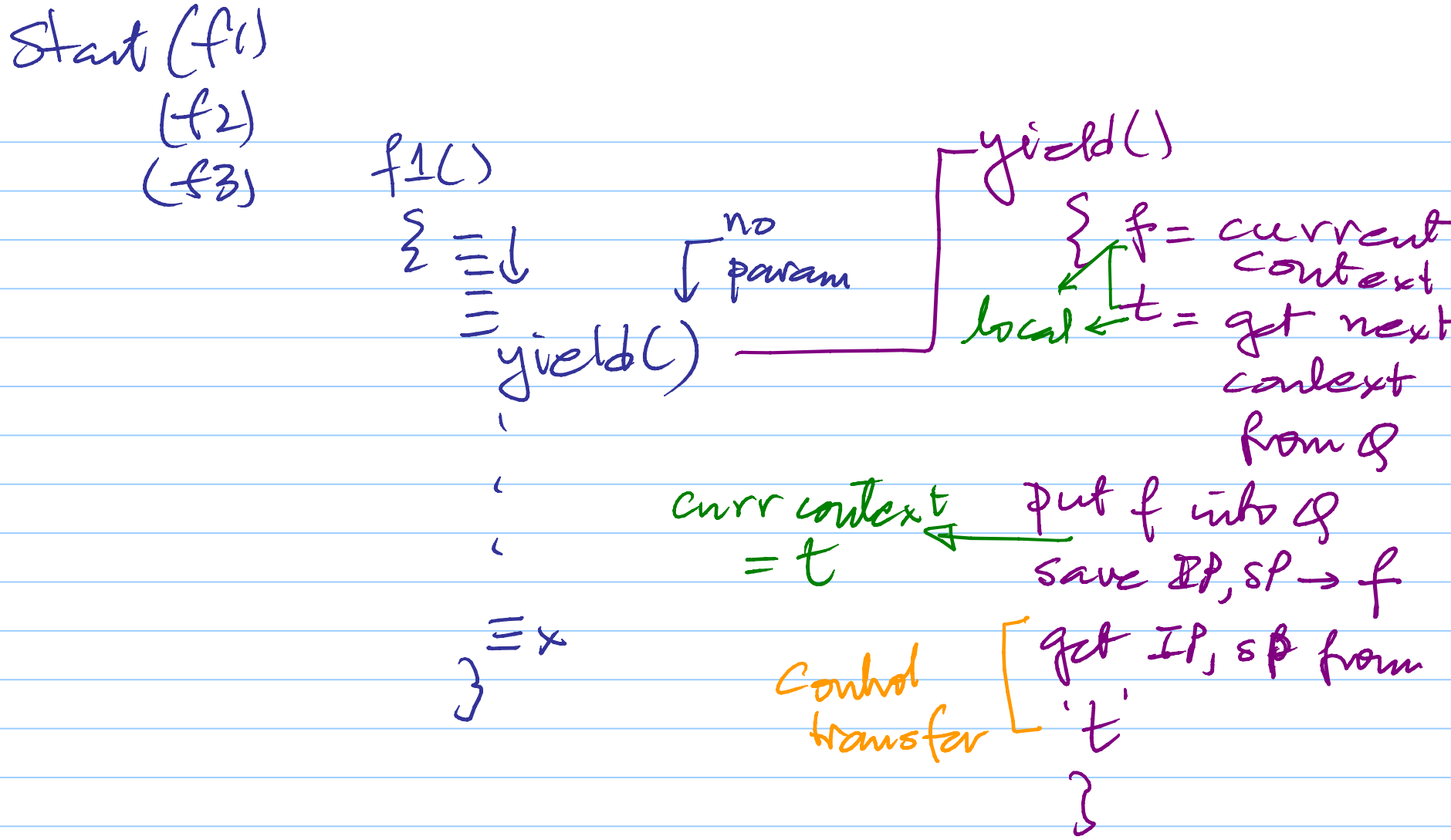
↳ FIFO scheduler

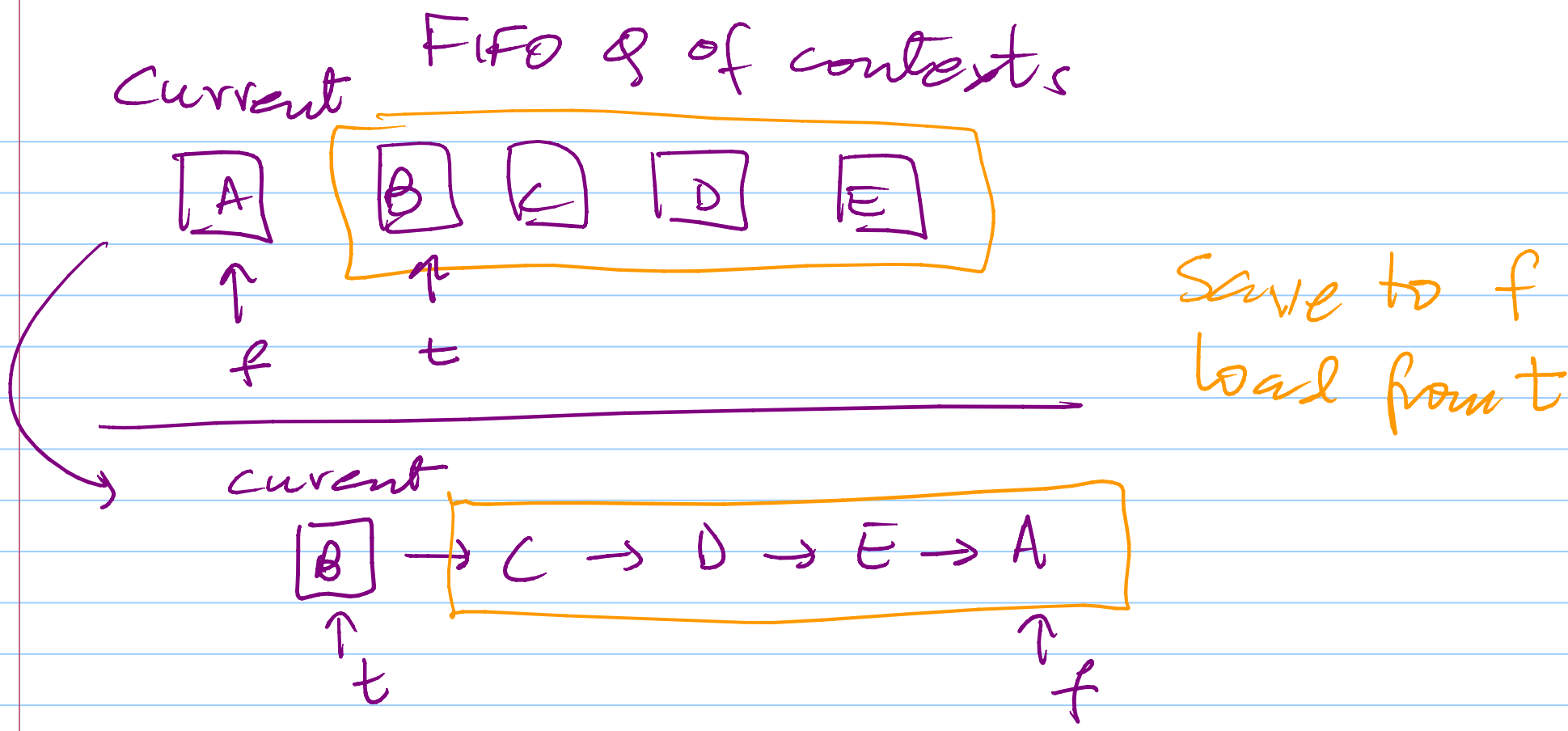
startthread (f)

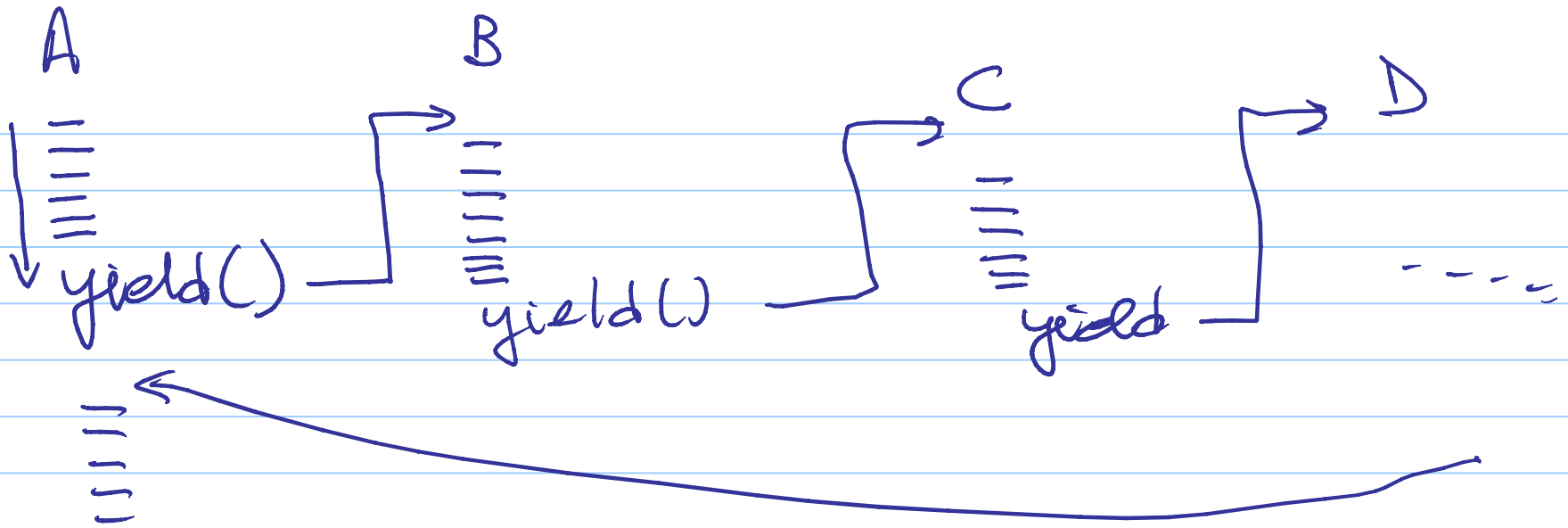
runthreads()

```
runthread()
{ get C from top of Q → current context
  switch to C (change IP, & SP)
  // never returns
}
```

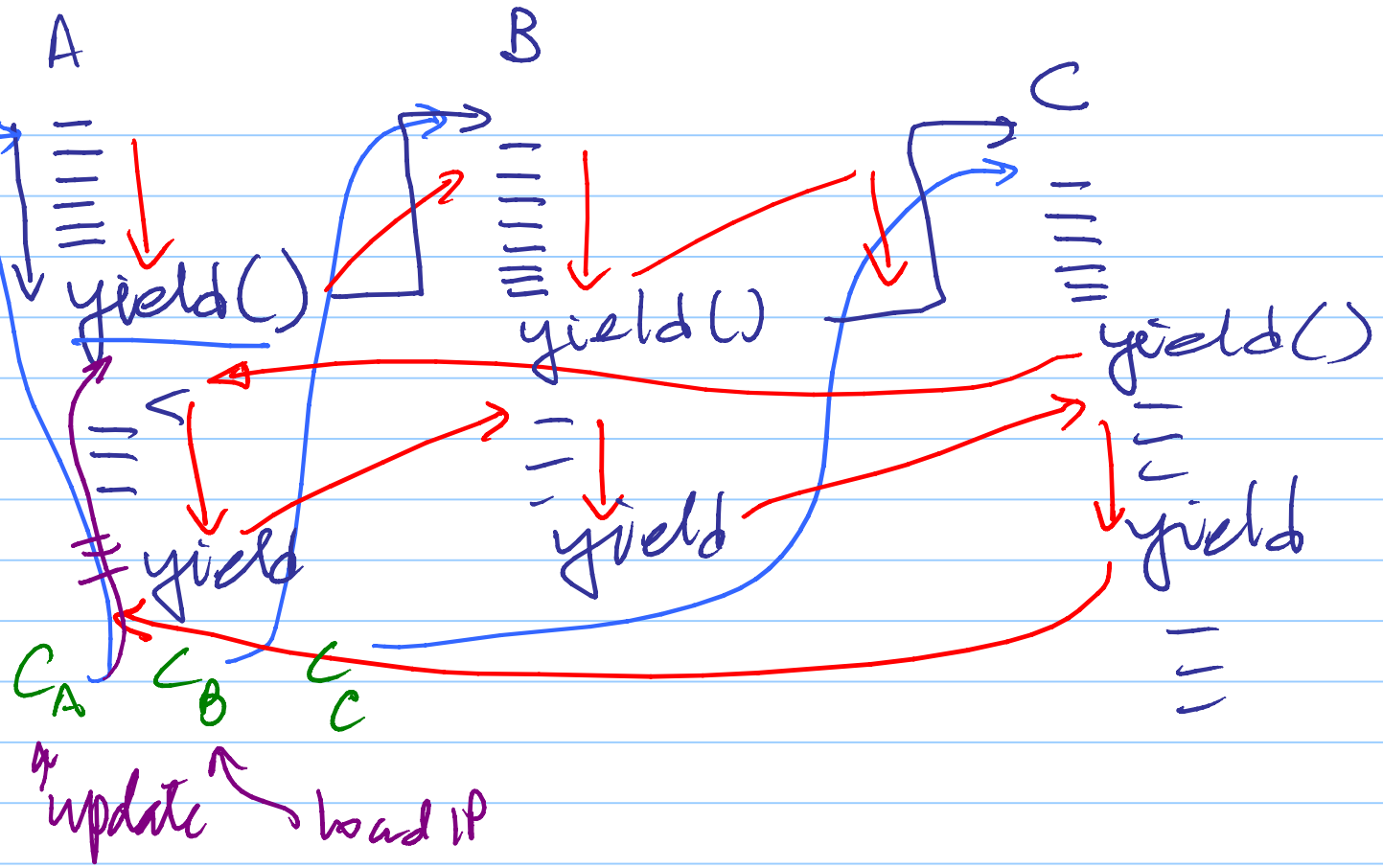
global


start (f1)
(f2)
(f3)
f1()
{
=
=
=
yield()
=
=
=
x
}
no param
yield()
{
f = current context
local t = get next context from Q
curr context = t
put f into Q
save IP, SP → f
get IP, SP from 't'
}
Could transfer


FIFO q of contexts
current
A
B
C
D
E
f
t
save to f
load from t
current
B
C
D
E
A
t
f


A
B
C
D
yield()
yield()
yield

start(A)
» (B)
» (C)
RUN



- Take a process → virtual CPU
  - ↳ main() of a C program
- Create thousands (coroutines)
- need interrupts? kernel?
  - → NO, if non preemptive
  - user level code