Exercise sheet 10 – Deadlock analysis

Goals:

- Deadlocks

Exercise 10.1: Deadlocks 1

The three processes `P1`, `P2`, and `P3` are executing the following code:

P1	P2	P3
P(S)	P(T)	P(U)
P(U) <=	. <=	P(S) <=
work_with_s_and_u();	work_with_t();	work_with_s_and_u();
V(S)	V(T)	V(U)
V(U)		V(S)

All semaphores start with the value 1; the arrow shows the code which is executed at the moment.

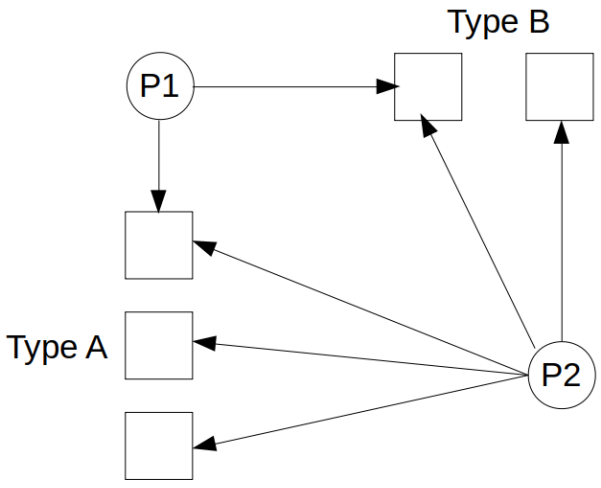
(a) Draw a system resource acquisition graph for this situation!

(b) Show that a deadlock exists.

(c) Show two possibilities to avoid the deadlock!

Exercise 10.2: Deadlocks 2

Look at this system resource acquisition graph:



(a) Is there a deadlock?

(b) Is the state safe?

(c) Find a sequence of operations which would cause a deadlock!

(d) Is it allowed to fulfil the request of `P2` for the two resources of type B?

Exercise 10.3: Deadlocks behaviour

(a) What happens on a deadlock on a desktop system?

(b) What happens on a deadlock on a server system?

(c) What happens on a deadlock on a smartphone?

(d) What happens on a deadlock on a safety critical realtime system (e.g. in a car)?

Exercise 10.4: Deadlocks analysis on existing C code

(a) Update the `OS_exercises` repository with `git pull`.

(b) Change into the `OS_exercises/sheet_10_deadlocks/deadlock_code_analysis` directory.

(c) Inspect the `deadlock_analysis.c`.

(d) Build and run the program.

(e) Does the program work correctly? Is there an error?

(f) Try to analyse the behaviour.

(g) Fix the bug.

(h) Build and run the program.