

Dokumentation:

1. Wenn ein Prozess, das Kinder hat, exits, dann werden die Kinder dem initialen Prozess (a) am Schluss zugewiesen. Hat Prozess b, Kindprozess c und c hat Kinderprozess d. Beim exit vom b, werden c und d getrennt an a zugewiesen.

```
vladb@VladB ~/G/h/S/B/H/HW2_process-api (master)> ./fork.py -s 30 -c
```

```
ARG seed 30
```

```
ARG fork_percentage 0.7
```

```
ARG actions 5
```

```
ARG action_list
```

```
ARG show_tree False
```

```
ARG just_final False
```

```
ARG leaf_only False
```

```
ARG local_reparent False
```

```
ARG print_style fancy
```

```
ARG solve True
```

Process Tree:

a

```
Action: a forks b
```

a

└─ b

```
Action: b forks c
```

a

└─ b

└─ c

```
Action: a forks d
```

a

├─ b

└─┬─ c

└─ d

```
Action: c forks e
```

a

Action: c EXITS

```
└─ b
|   └─ c
|       └─ e
└─ d
```

```
a
└─ b
└─ d
└─ e
```

2. Mit 0.1 wird am Schluss nur a bleiben. Mit 0.9 sind es sehr viele Prozesse mit mehreren Ästen.

```
vladb@VladB ~/G/h/S/B/H/HW2_process-api (master)> ./fork.py -a 10 -f 0.1 -c
```

ARG seed -1

ARG fork_percentage 0.1

ARG actions 10

ARG action_list

ARG show_tree False

ARG just_final False

ARG leaf_only False

ARG local_reparent False

ARG print_style fancy

ARG solve True

Process Tree:

```
a
```

Action: a forks b

```
a
└─ b
```

Action: b EXITS

```
a
```

Action: a forks c

```
a
```

```

└─ c

Action: c EXITS

a

Action: a forks d

a
└─ d

Action: d EXITS

a

Action: a forks e

a
└─ e

Action: e EXITS

a

Action: a forks f

a
└─ f

Action: a forks g

a
├─ f
└─ g

```

```

vladb@VladB ~/G/h/S/B/H/HW2_process-api (master)> ./fork.py -a 10 -f 0.9 -c

```

```

ARG seed -1
ARG fork_percentage 0.9
ARG actions 10
ARG action_list
ARG show_tree False
ARG just_final False
ARG leaf_only False
ARG local_reparent False
ARG print_style fancy
ARG solve True

```

Process Tree:

```

a

```

Action: a forks b

```
a
└─ b
```

Action: b forks c

```
a
└─ b
   └─ c
```

Action: c forks d

```
a
└─ b
   └─ c
      └─ d
```

Action: b forks e

```
a
└─ b
   └─ c
      └─ d
   └─ e
```

Action: b forks f

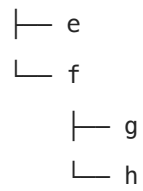
```
a
└─ b
   └─ c
      └─ d
   └─ e
   └─ f
```

Action: f forks g

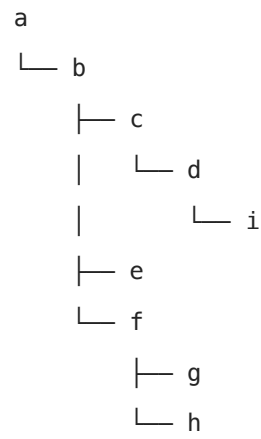
```
a
└─ b
   └─ c
      └─ d
   └─ e
   └─ f
      └─ g
```

Action: f forks h

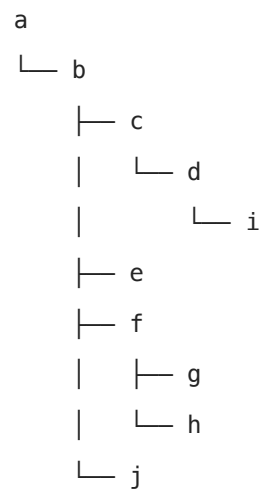
```
a
└─ b
   └─ c
      └─ d
   └─ h
```



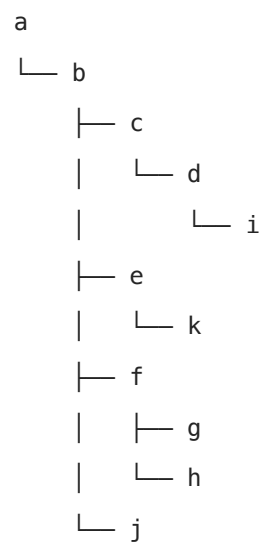
Action: d forks i



Action: b forks j



Action: e forks k



3. -t zeigt die jeweiligen Graphen aber nicht die Aktionen. Man sollte in der Lage sein jedes Mal die Aktion richtig zu vermuten.

```
vladb@VladB ~/G/h/S/B/H/HW2_process-api (master)> ./fork.py -t
```

```
ARG seed -1
```

```
ARG fork_percentage 0.7
```

```
ARG actions 5
```

```
ARG action_list
```

```
ARG show_tree True
```

```
ARG just_final False
```

```
ARG leaf_only False
```

```
ARG local_reparent False
```

```
ARG print_style fancy
```

```
ARG solve False
```

Process Tree:

a

Action?

a

└─ b

Action?

a

└─ b

└─ c

Action?

a

└─ b

└─┬─ d

└─ c

Action?

a

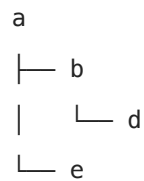
└─ b

└─┬─ d

└─ c

└─ e

Action?



4. siehe 1. Mit -R werden die Kinder nicht dem initialen Prozess zugewiesen, sondern dem lokalen Elternprozess

```
vladb@VladB ~/G/h/S/B/H/HW2_process-api (master)>
./fork.py -A a+b,b+c,c+d,c+e,c- -c
```

```
ARG seed -1
ARG fork_percentage 0.7
ARG actions 5
ARG action_list a+b,b+c,c+d,c+e,c-
ARG show_tree False
ARG just_final False
ARG leaf_only False
ARG local_reparent False
ARG print_style fancy
ARG solve True
```

Process Tree:

```
a
```

Action: a forks b

```
a
└─ b
```

Action: b forks c

```
a
└─ b
   └─ c
```

Action: c forks d

```
a
└─ b
   └─ c
      └─ d
```

```

                                └─ d

Action: c forks e

a
└─ b
   └─ c
      └─ d
         └─ e

```

Action: c EXITS

```

a
└─ b
   └─ d
      └─ e

```

```

vladb@VladB ~/G/h/S/B/H/HW2_process-api (master)>
./fork.py -A a+b,b+c,c+d,c+e,c- -R -c

```

ARG seed -1

ARG fork_percentage 0.7

ARG actions 5

ARG action_list a+b,b+c,c+d,c+e,c-

ARG show_tree False

ARG just_final False

ARG leaf_only False

ARG local_reparent True

ARG print_style fancy

ARG solve True

Process Tree:

```

a

Action: a forks b

a
└─ b

Action: b forks c

a
└─ b

```



```

          └─ c

Action: c forks d

a
└─ b
  └─ c
    └─ d

Action: c forks e

a
└─ b
  └─ c
    └─ d
      └─ e

Action: c EXITS

a
└─ b
  └─ d
    └─ e

```

5. -F zeigt nur am Ende den Graph. Durch die Aktionen sollte man schon in der Lage sein, den Baumgraph zu zeichnen.

```
vladb@VladB ~/G/h/S/B/H/HW2_process-api (master)> ./fork.py -F
```

```

ARG seed -1
ARG fork_percentage 0.7
ARG actions 5
ARG action_list
ARG show_tree False
ARG just_final True
ARG leaf_only False
ARG local_reparent False
ARG print_style fancy
ARG solve False

```

Process Tree:

```
a
```

```
Action: a forks b
Action: b forks c
Action: c EXITS
Action: b forks d
Action: a forks e
```

Final Process Tree?

6. In Fällen wo nur geforkt wurde, kann man immer sagen wie es geschehen war. Also die Anzahl der Prozesse am Schluss ist gleich der n Aktionen + 1 (a gab ja schon) ist. Ist nach 5 Aktionen nur zwei Prozesse übrig geblieben, ist die Lösung nicht eindeutig!

```
vladb@VladB ~/G/h/S/B/H/HW2_process-api (master)> ./fork.py -t -F -s 5
```

```
ARG seed 5
ARG fork_percentage 0.7
ARG actions 5
ARG action_list
ARG show_tree True
ARG just_final True
ARG leaf_only False
ARG local_reparent False
ARG print_style fancy
ARG solve False
```

Process Tree:

a

```
Action?
Action?
Action?
Action?
Action?
```

Final Process Tree:

a

└ d

```
vladb@VladB ~/G/h/S/B/H/HW2_process-api (master)> ./fork.py -t -F -s 3
```

ARG seed 3

ARG fork_percentage 0.7

ARG actions 5

ARG action_list

ARG show_tree True

ARG just_final True

ARG leaf_only False

ARG local_reparent False

ARG print_style fancy

ARG solve False

Process Tree:

a

Action?

Action?

Action?

Action?

Action?

Final Process Tree:

a

├ b

| └ c

| └ f

└ d

└ e