HOMEWORK 4

Wu Jiayao 517370910257

EX.1

1.

All threads might be blocked by this interrupt. Use kernel-level threads, or mix kernel-level and user-level threads together.

2.

It can be implemented as follow. Before interrupt happens, just give an alarm clock to every thread.

EX.2

When a process is blocked by the **waituntil** operation, the system has to keep checking the value of the bool variable, which is a waste of computing resources. And this solution can not be applied in more complext situation where there are more than two processes or various rules of blocking.

EX.3

3.1

```
#!/bin/bash
if [ ! -f "ex3_1.txt" ]; then
    echo 0 >ex3_1.txt
fi
for ((i = 0; i < 100; ++i)); do
    end=$(tail -1 ex3_1.txt)
    end=$((end + 1))
    echo $end >>ex3_1.txt
done
```

Run the script as

```
./ex3_1.sh & ./ex3_1.sh
```

The race condition starts at the second line, where 1 appears twice in Line 2 and 3.

3.2

```
#!/bin/bash
if [ ! -f "ex3_2.txt" ]; then
    echo 0 >ex3_2.txt
fi
for ((i = 0; i < 100; ++i)); do
    (
        flock 5
        end=$(tail -1 ex3_2.txt)
        end=$((end + 1))
        echo $end >>ex3_2.txt
    ) 5<>ex3_2.txt
done
```

Run the script as

```
./ex3_2.sh & ./ex3_2.sh
```

EX.4

```
gcc -pthread -o ex4 /src/ex4.c
```