To do:
Sort from most time used to least time used.
Create a state system/view to easily understand the data.
Implement a more efficient searching/sorting algorithm.
Clean up the code.

Code:

```
loadModule("/TraceCompass/Trace");

//get the active trace
var trace = getActiveTrace();
if(trace==null){
	print("No trace is active.");
	exit();
}

//this list will be used like a 2d array in java
//it will contain one list for each CPU of the "sched_switch" events
var cpus = [];

//this block will create the 2d list mentioned above
var iter = getEventIterator(trace);
var event = null;
while (iter.hasNext()){
	event = iter.next();

	var eventName = event.getName();
	var eventCPU = getEventFieldValue(event,"CPU")

	if(eventName=="sched_switch"){
		//create a new CPU list if this is the first event for that CPU
		if(cpus[eventCPU]==null){
			var events = [];
			events[0] = event;
			cpus[eventCPU] = events;
		//otherwise add the event to the end of the existing CPU list
		}else{
			cpus[eventCPU][cpus[eventCPU].length] = event;
		}
	}
}

//this block calculates, for each CPU, the time from the 'i'th sched_switch event to the 'i+1'th and
matches that time with the corresponding thread id
for(i=0; i<cpus.length; i++){
	var switches = [];
	var durations = [];
	var p = 0;
```

```
	var prev = -1;
	print("CPU " + i);
	for(j=0; j<cpus[i].length; j++){
		if(prev!=-1){
			switches[p] = cpus[i][j];
			durations[p] = cpus[i][j].getTimestamp().toNanos() - prev;
			p++;
		}
		prev = cpus[i][j].getTimestamp().toNanos();
	}
	addDurations(switches,durations);
}

//done
print("Done");

//this function matches, for each CPU, the thread id's together and adds their durations
function addDurations(events,durations){
	var newListEvents = [];
	var newListDurations = [];
	var q = 0;
	for(k=0; k<events.length; k++){
		var exists = false;
		for(l=0; l<newListEvents.length; l++){
			//if the thread is already in the new list, add the additional duration to the
existing duration

	if(getEventFieldValue(events[k],"prev_tid")==getEventFieldValue(newListEvents[l],"prev_tid")){
					newListDurations[l] = newListDurations[l] + durations[k];
					exists = true;
			}
		}
		//if the thread is not yet represented in the new list, add it
		if(!exists){
			newListEvents[q] = events[k];
			newListDurations[q] = durations[k];
			q++;
		}
	}

	//print the calculated data
	for(k=0; k<newListEvents.length; k++){
		printEvent(newListEvents[k], newListDurations[k]);
	}
}

//this function prints the data to the console
```

```
function printEvent(event, duration){
        print(getEventFieldValue(event,"prev_tid") + ":" + getEventFieldValue(event,"prev_comm") + " -
-> " + duration + " nanoseconds");
}
```

Output: