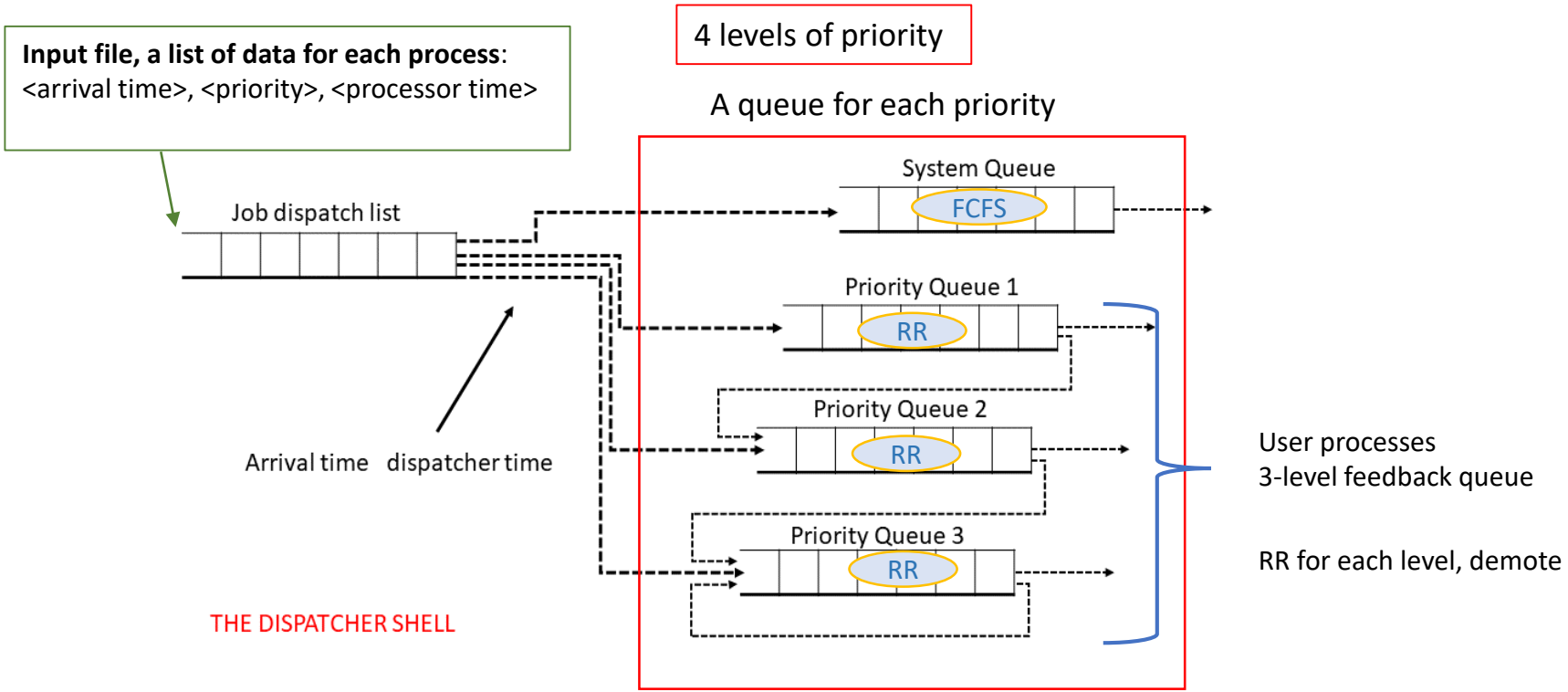

Input file, a list of data for each process:
<arrival time>, <priority>, <processor time>
4 levels of priority
A queue for each priority
System Queue
FCFS
Job dispatch list
Priority Queue 1
RR
Priority Queue 2
RR
Priority Queue 3
RR
Arrival time
dispatcher time
THE DISPATCHER SHELL
User processes
3-level feedback queue
RR for each level, demote

Dispatcher and the process

```
gcc -g dispatcher.c -o dispatcher -Wall
gcc -g sigtrap.c -o process -Wall
```

- Sigtrap.c compile to an executable, name it "`process`" – see the make file
- `./process` can run, it will stop after 20 sec

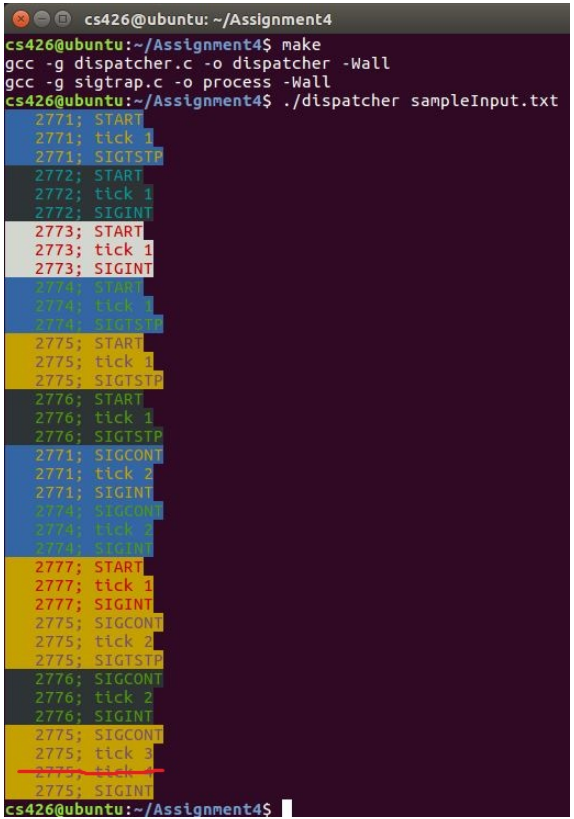
`./process` prints out related scheduling events.

- Your dispatcher will read from an **Input file**. The file contains a list of data for each process : <arrival time>, <priority>, <processor time>, add to job dispatch list (a queue).
- Your dispatcher will insert a job into a priority queue at its arrival time, it will be scheduled accordingly.
- Your dispatcher picks from a queue the process scheduled to run, creates a child process by forking a process and loading the precompiled `process`. So this `process` (with own pid) will be running.

- When the `process` is running, the scheduling events will be print out by the `process`.

Dispatcher and the process

- sigtrap - report system signals applied to process
  - usage: sigtrap [n]
  - [n] is time for process to exist - default 20 seconds
- program ticks away reporting process id and tick count every second.
  - the program traps and reports the following signals:
  - SIGINT, SIGQUIT, SIGHUP, SIGTERM, SIGABRT, SIGCONT, SIGTSTP
  - Signals are set when the dispatcher acts upon the process by sending signal using `kill (..,..)` statement.
- program can not trap SIGSTOP or SIGKILL
- to help identify specific processes, the program uses the process id to select one of 32 colour combinations for the display to an ASCC terminal.
- output is to stdout (set in #define), reset to BLACK and NORMAL and flushed after every printf.

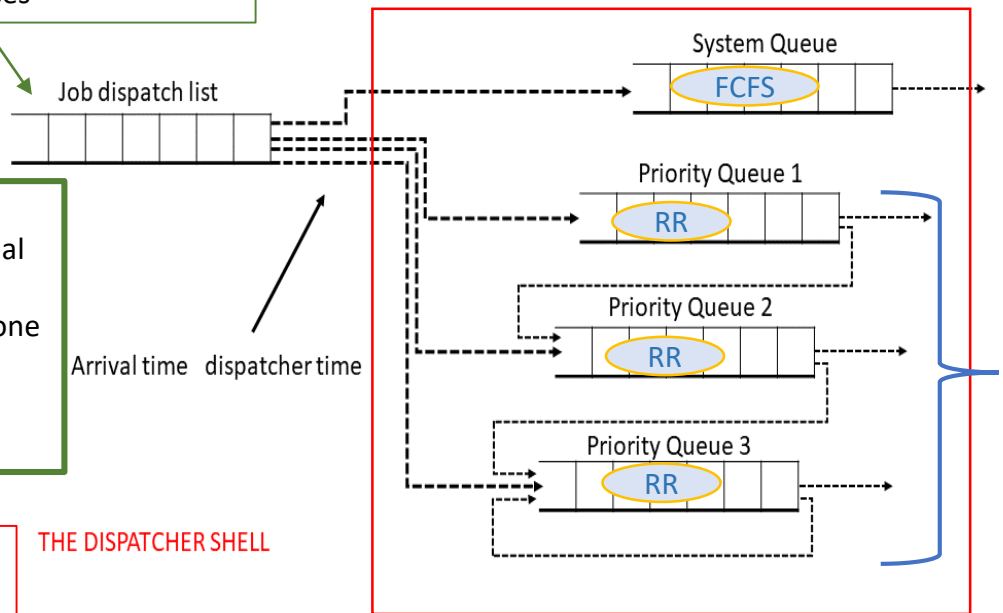


A data structure for a process

Input file, a list of data for each process:
<arrival time>, <priority>, <processor time>
A queue for all the processes

4 levels of priority

A queue for each priority



Arrival time dispatcher time

Simulate time ticks in a loop
a) start with processing potential new arrivals for this time tick;
b) go through the queues find one process to run, i.e. create a process to run the real process
c) And, other operations…

This will involve a real process (sigtrap.c)

THE DISPATCHER SHELL

Operations on process

startProcess
suspendProcess
restartProcess
terminateProcess

User processes
3-level feedback queue

RR for each level, demote