1 DB stuff

1.

2 Content of configuration-file

1. host = the IP of the MySQL-server
2. username =
3. password =
4. database = name of database that contains the EDACC-tables
5. experiment = the ID of the experiment to be launched
6. gridqueue = the ID of the Queue for which this package was generated. This is important because it influneces the PBS-script content.

3 Content of the package for the grid-queues and the structure

Everything is contained in a directory called: /EDACC_[experiment_id]

1. /configuration-file
2. /solver/[binary of solvers]
3. /pbs-skript
4. /client-binary
5. /instances/[instances]
6. /results/
7. /run.sh

Convention regarding the name of the result file in table Experiment Results:

- resultFileName= ¡ExperimentID¿_¡SolverConfigID¿_¡InstanceID¿_¡run¿.res
- resultFileName.client = output of the client for that job
- resultFileName.solver = output of the solver for that job

Status of the job (table: ExperimentResults) to be replaced

-3: client crash

-2: solver crash

-1: not started

0: running

1: finished normaly by solver

2: terminated by ulimit maxtime

3: terminated by ulimit maxmem

Status-codes for a job new version

-5: launcher crash

-4: watcher crash

-3: solver crash

-2: verifier crash

-1: not started

0: running

1: finished normally by solver without exceeding any limit

2_: finished by exceeding some limits (here further extension are possible with the actual limit)

Result-codes for a job new version
All codes starting with a 1 are correct and certified results.

11 SAT

10 UNSAT

0 UNKNOWN

-1 wrong answer

-2 limit exceeded

- -21 cpu-time
- -22 wall clock-time
- -23 memory
- -24 stack-size
- -25 output-size

-3xx received signal:xx stands for the signal-code (see appendix A)

Color codes of job status in GUI experiment browser:

red: -2 an error occurred

blue: -1 not started

orange: 0 running

green: 1,2,3 finished

4 Parameters

Predefined parameter names:

seed values for seeds are generated by the job generator.

instance the instance for the solver.

5 Content launcher-output

- Environment variables like data about:
  - grid, node, place time
  - CPU-props, MEM-props.
  - PBS-variables if any
- Job-details
  - Experiment infos
  - job infos: solver, parameters (configuration), instances
  - limits set for job
  - output-name and location of the job

Further details from the client that are needed only for debugging purposes will be written to a file called: client_pidOfClient_node.cout. This file will contain information regarding:

- status and all operations for the DB-connection
- presence of the needed files on the system
- and file-system activities like writing solvers and instances on the file-system or writing from the file-system to the DB.

6 Content watcher-output

see run-solver

7 verifier-output

arbitrary : to be defined

A Linux Signals

Signal Name	Number	Description
SIGHUP	1	Hangup (POSIX)
SIGINT	2	Terminal interrupt (ANSI)
SIGQUIT	3	Terminal quit (POSIX)
SIGILL	4	Illegal instruction (ANSI)
SIGTRAP	5	Trace trap (POSIX)
SIGIOT	6	IOT Trap (4.2 BSD)
SIGBUS	7	BUS error (4.2 BSD)
SIGFPE	8	Floating point exception (ANSI)
SIGKILL	9	Kill(can't be caught or ignored) (POSIX)
SIGUSR1	10	User defined signal 1 (POSIX)
SIGSEGV	11	Invalid memory segment access (ANSI)
SIGUSR2	12	User defined signal 2 (POSIX)
SIGPIPE	13	Write on a pipe with no reader, Broken pipe (POSIX)
SIGALRM	14	Alarm clock (POSIX)
SIGTERM	15	Termination (ANSI)
SIGSTKFLT	16	Stack fault
SIGCHLD	17	Child process has stopped or exited, changed (POSIX)
SIGCONT	18	Continue executing, if stopped (POSIX)
SIGSTOP	19	Stop executing(can't be caught or ignored) (POSIX)
SIGTSTP	20	Terminal stop signal (POSIX)
SIGTTIN	21	Background process trying to read, from TTY (POSIX)
SIGTTOU	22	Background process trying to write, to TTY (POSIX)
SIGURG	23	Urgent condition on socket (4.2 BSD)
SIGXCPU	24	CPU limit exceeded (4.2 BSD)
SIGXFSZ	25	File size limit exceeded (4.2 BSD)
SIGVTALRM	26	Virtual alarm clock (4.2 BSD)
SIGPROF	27	Profiling alarm clock (4.2 BSD)
SIGWINCH	28	Window size change (4.3 BSD, Sun)
SIGIO	29	I/O now possible (4.2 BSD)
SIGPWR	30	Power failure restart (System V)

Tabelle 1: Linux Signals

As noted above, processes can ignore, block, or catch all signals except SIGSTOP and SIGKILL. If a process catches a signal, it means that it includes code that will take appropriate action when the signal is received. If the signal is not caught by the process, the kernel will take default action for the signal.