CS2103 PROJECT MANUAL

Command line task manager

He Haocong	**Liu Jialong**	**Wang Xiangyu**	**Zhou Biyan**
U099121H	U099122U	U099120W	U094837M

November 4, 2010

Contents

1 Introduction

abc abc

2 User Guide

2.1 Quick Start

This section introduces you the minimum amount of commands to get started.

1. Start taskManager shell

   On Mac OS and GNU Linux (referred to as "*nix" later), taskManager can be started from shell by commands:

   ```
   $ cd the/folder/containing/taskManager
   $ ./taskManager
   ```
   [1]

   On Microsoft Windows, taskManager can be launched in command prompt as well, or simply by double clicking.

   Once taskManager is launched, you will see a prompt like ">_", and you will start typing commands!

2. Add some tasks

   ```
   $ ./taskManager
   > add "Sample task 1"
   > add "Sample task 2"
   TaskManager:  This task is highly similiar to some existing
   task, do you really want to add it?  y
   ```

   To add a task, simply use `add` command followed by the description of the task in a pair of quotation marks.

   If no error messages are shown, the task is successfully added. TaskManager may prompt for confirmation if the task to be added is highly similar to some existing task(s) to help prevent people forget adding tasks, which is the case in the example above.

3. List the existing tasks

[1] *nix version can be run at any directory after installing taskManger – "make install". See details in section "Compilation and Installation".

```
> ls
1 Sample task 1
2 Sample task 2
```

To see the existing tasks, use `ls` command. By default, the taskManager shows the serial numbers and the descriptions of the tasks.

4. Mark a task as finished

   ```
   > finish 2
   > ls
   1 Sample task 1
   2 f Sample task 2
   ```

   To finish an existing task, use the finish command followed by the serial number of the task to finish. Notice that for finished task, an 'f' is shown between serial number and task description.

5. Remove task(s)

   ```
   > ls
   1 f Sample task 1
   2 Sample task 2
   > rm 1
   TaskManager:  Do you really want to remove this task permanently?y
   > ls
   2 Sample task 2
   ```

   To remove an existing task, use `rm` command followed by the serial number of the task to remove. TaskManager will prompt for confirmation when removing tasks.

6. Exit from taskManager

   ```
   > exit
   ```

   To quit from taskManager, use `exit` command. All changes to the existing tasks will be automatically saved.

2.2 More commands

2.2.1 read, import and export

TaskManager stores the tasks in an XML file which is by default `~/record.xml` on *nix, and `%USERPROFILE%\record.xml` on Windows.

TaskManager also supports importing/exporting the existing tasks from/to XML and HTML files. This is done by `read`, `import` and `export` commands.

`read` reads an XML file, list all the tasks it contains without affecting the current task list.

This is helpful when you only want to peek the content of an xml file without really importing it.

```
> ls
1 Sample task 2
> read thisweek.xml [2]
1 f CS2103 midTerm Sep 29 06:30 - 07:30 pm MPSH 1B
2 f CS3230 midTerm Oct 15 06:00 pm
3 f CS3241 midTerm Oct 07 lecture
4 f CS3244 midTerm Oct 04 lecture
5 f ST2132 midTerm Oct 08 LT33 12:15 - 1:30 pm
> ls
1 Sample task 2
```

`import` is similar to `read` command. It reads the content of the XML file and appends all the tasks in it to current task list.

```
> ls
1 Sample task 2
> import mytasks.xml
1 Sample task 2 [3]
2 f CS2103 midTerm Sep 29 06:30 - 07:30 pm MPSH 1B
3 f CS3230 midTerm Oct 15 06:00 pm
4 f CS3241 midTerm Oct 07 lecture
5 f CS3244 midTerm Oct 04 lecture
6 f ST2132 midTerm Oct 08 LT33 12:15 - 1:30 pm
```

`export` exports the current task list to an XML or HTML file.

```
> export sampletasks.xml [4]
> exit
$ cat sampletasks.xml
<taskList>
<task>
<serialNumber> 1 </serialNumber>
<deadline> 1288473083 </deadline>
```

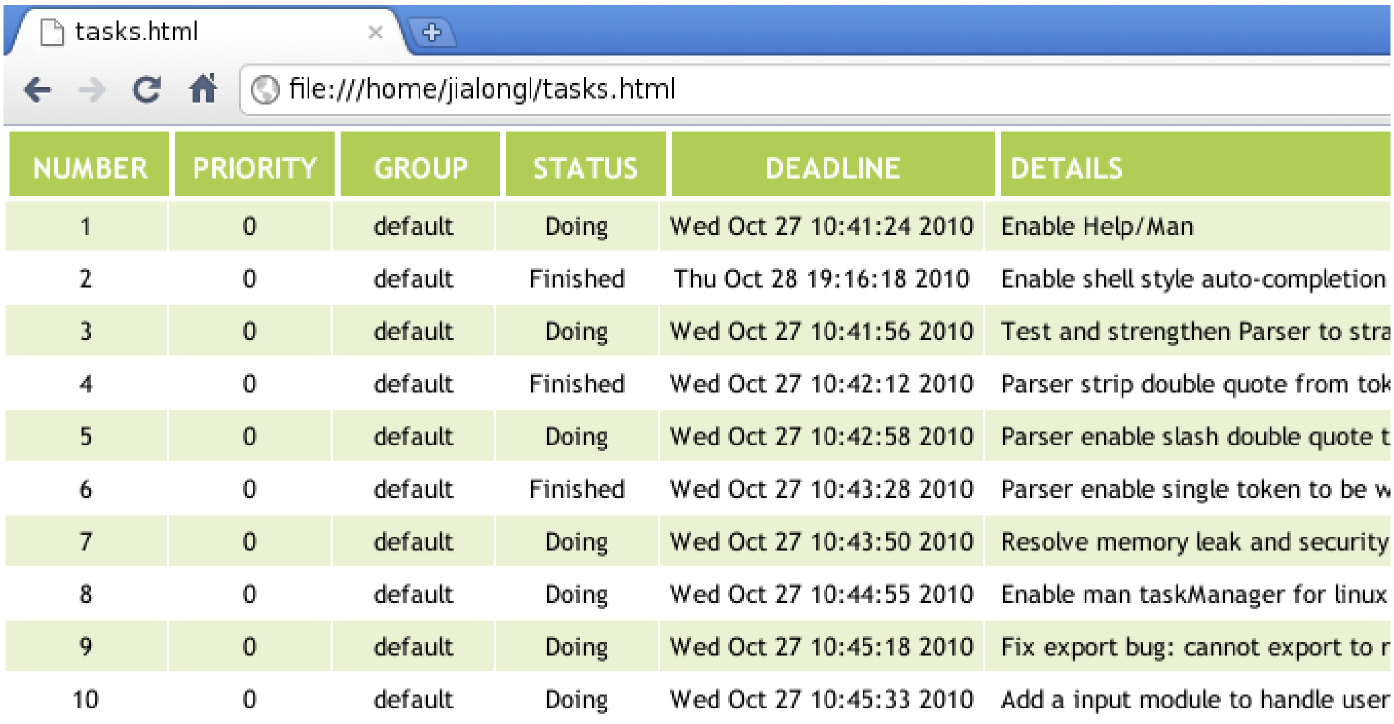
[2] The file name is not quoted. If the file name contains space, please quote it with a pair of quotation marks.

[3] Task 1 is still in task list. Importing tasks will not erase existing tasks.

```
<priority> 0 </priority>
<description> Sample task 2 </description>
<group> default </group>
<isFinished> 0 </isFinished>
</task>
</taskList>
```

`export` can also be used to generate an HTML file which is more visually pleasant in your favourate browser.

```
> export -html sampletasks.html
> exit
```



NUMBER	PRIORITY	GROUP	STATUS	DEADLINE	DETAILS
1	0	default	Doing	Wed Oct 27 10:41:24 2010	Enable Help/Man
2	0	default	Finished	Thu Oct 28 19:16:18 2010	Enable shell style auto-completion
3	0	default	Doing	Wed Oct 27 10:41:56 2010	Test and strengthen Parser to stra
4	0	default	Finished	Wed Oct 27 10:42:12 2010	Parser strip double quote from tok
5	0	default	Doing	Wed Oct 27 10:42:58 2010	Parser enable slash double quote t
6	0	default	Finished	Wed Oct 27 10:43:28 2010	Parser enable single token to be w
7	0	default	Doing	Wed Oct 27 10:43:50 2010	Resolve memory leak and security
8	0	default	Doing	Wed Oct 27 10:44:55 2010	Enable man taskManager for linux
9	0	default	Doing	Wed Oct 27 10:45:18 2010	Fix export bug: cannot export to r
10	0	default	Doing	Wed Oct 27 10:45:33 2010	Add a input module to handle user

Figure 1: tasks exported as webpage [5]

[4] Currently `export` does not support environmental variables in path. E.g. `export ~/abc.xml` will not export the file to user's home directory /home/username/.

[5] Page may not render correctly in IE 6 or its earlier versions.

2.2.2 task

```
> task 1
  Number:  1   Deadline:Sun Oct 31 05:11:23 2010
  Priority:0   Status:  Doing
  Group:   default
  Details:
    Sample task 2 [6]
```

To show detail information of a task, use `task` command followed by serial number of the task.

2.2.3 pri

```
> pri 1 10
> task 1
  Number:  1   Deadline:  Sun Oct 31 05:11:23 2010
  Priority:10  Status:    Doing
  Group:   default
  Details:
    Sample task 2
```

To change the priority of a task, use `pri` command followed by the serial number of a task and its new priority. Priority is typically a number between -20 and 20. By default, the priority of a newly added task is 0.

2.2.4 edit

```
> edit 1 -d "Sample task 3" -p 12 -t 1d[7] -g SampleGroup -f yes
> task 1
  Number:  1   Deadline:  Mon Nov 1 05:47:59 2010
  Priority:12  Status:    Finished
  Group:   SampleGroup
  Details:
    Sample task 3
```

To edit a task, `edit` command is used like this:
`edit taskSerialNumber -d newDescription -p newPriority -t newDeadline -g newGroup -f finished_or_not`

[6] Adding tasks with detailed information is covered in section 2.3.1. In this example, default values are shown.

Only serial number is compulsory. Besides, to finish a task, `finish 1` is equivalent to `edit 1 -f yes`.

2.2.5 undo and redo

> `undo`

Undo the last command. Note that it has no effect on commands like `ls`, `export`, `tui` and `undo`.

> `redo`

Redo the last undo. It can be executed util all the undo's are re-done.

2.3 Using Options

Like `edit`, some of the commands come with options to support more functionality. In this section they are introduced in great detail.

2.3.1 add

`-t` add a task with a deadline:

```
> add "some task" -t 3d2h
> add "some task" -t b2d
> add "some task" -t 12345
```

taskManager support 3 types of time format:

`"plus" format` "Plus" format specifies the how much time left for the task, and it has form ?w?d?h?m[8], where each question mark stands for a number (not a digit).
For example, 3d2h means the task will due after 3 days 2 hours the moment the command is executed – you have 3 days and 2 hours to finish it.

`"by" format` "By" format has a similar form of "plus" format. It is in the form of b?w?d?h?m[9], where each question mark stands for a number (not a digit). For example:

[7] -t 1d means setting the deadline to be 1 day later. Time formats that taskManager accept are discussed in section 2.3.1.

[8] At least one of letters w/d/h/m should be specified.

`b0d22h`	by 10:00pm today.
`b2d`	by 23:59 tomorrow.
`b1w`	by the end of this week. i.e. 23:59 of the coming Sunday.
`b0w5d`	by 23:59 of the coming Friday.
`b2w3d8h`	by 8:00am of the next next Wednesday.

`Unix timestamp` "Unix timestamp" means the number of seconds elapsed since Jan 1, 1900 00:00:00. It is not recommended for users, but rather used as the lower-order method for developers.

`-p` add a task with a priority:

```
> add "some important task" -p 20
```

`-g` specify a group for a task:

```
> add "the task with group"[10] -g SampleGroup
```

Options are not compulsory. Different options can be used together. For example `add "some task" -p 10 -g "special task" -t 4d`.

2.3.2 ls

`-s` sorts the existing tasks:

```
> ls -s "deadline priority"
```

A more general format is: > `ls -s "keyword1 keyword2 ..."`

The listed tasks will be sorted by keyword1 then keyword2 . . .

Available search keywords are: deadline, priority and serialnumber. Prefix of a keyword is also acceptable. e.g. -s "p" will sort the list by priority.

Examples:

```
> ls -s "p d"
1 task 1 highest priorty.   3 Sun Oct 31 06:49:09 2010
2 task 2 high priority.     2 Mon Nov 1 06:54:42 2010
3 task 3 default priority.  0 Tue Nov 2 06:54:37 2010
```

`-k` filters tasks with a keyword[11]:

[9] At least one of letters w/d/h/m should be specified.

[10] If group name contains spaces, use a pair of quotation marks to quote it.

`ls -k *Sam?le*task`, where ? means any single character, * means any string. (including empty string).

For example, “This is a sample with a important task” will match *Sam?le*task as the first * matches “This is a ”, ? matches ‘p’ and the second * matches “ with a important ”.

“samqleTask” will also match *sam?le*task by letting both * to be empty string and ? to be ‘q’.

`-f` shows finished/unfinished tasks:

`ls -f yes` shows only finished tasks.

`ls -f no` shows only unfinished tasks.

`-g` shows tasks of a specific group:

`ls -g SampleTask` makes tasks only from SampleTask group shown.

Tip: Different options can be used together. When more then one restrictive options are there, conjuction of these restrictions are used. e.g. `ls -g SampleTask -f y` will show tasks that are finished AND from “SampleTask” group.

2.3.3 rm

Use -g option to remove a group of tasks:
`rm -g SampleTask` removes the entire SampleTask group.
`rm` can be used removeS several tasks once as well. e.g. `rm 1 2 3` removes tasks 1, 2 and 3.

Notice: Commands like finish, rm, export, etc. do not support all task-selective options like -g -k -f. Executing these commands on a selected task set can be done with command piping, which is discussed in section 2.5.1.

2.4 Text Based Interactive User Interface

blah blah blah

2.5 Advanced Usages

2.5.1 command piping

```
> ls | rm
```

[11] keyword is case insensitive.

TaskManager supports command piping for most commands though it is a bit different from traditional Unix pipe. Piping means if one command selects some tasks, then the selected tasks will be passed to the next command as input. The tasks after the last command will be shown as output. Piping in taskManager is done with symbol '|'. When a pipe signs appear in a command, the smaller commands (separated by pipes) are executed one by one from left to right. For examples:

1. finish all tasks
   `ls | finish`
2. remove all finished tasks
   `ls -f yes | rm`
3. import from a file and replace current task list
   `ls | rm | import newTasks.xml`
4. import all CS2103 group tasks from a file
   `read newTasks.xml | ls -g CS2103 | add`
5. export all CS2103 related tasks to a html file
   `ls -k *CS2103* | export -html cs2103tasks.html`
6. show details of CS2103 tasks, sort by priority
   `ls -g CS2103 | sort "pri" | task`

2.5.2 command mapping

```
> map "ls" "ls -f no"
```

TaskManager supports custom command mapping/aliasing. General format of `map` is:

```
> map "new command" "original command"
```

A simple mapping is like the previous example. This maps "ls" to "ls -f no", which means hide finished tasks when listing. To retain the original `ls` command, map `ls` to something else. For example:

```
> map "lsa" "ls"
> map "ls" "ls -f no"
```
[12]

More complex mapping makes use of symbol $. There are two kinds of $ symbols:

`$0` matches all characters from the current position.
`$1`, `$2`, `$3` . . . corresponds to one token separated by spaces.

[12] The order of mapping matters as commands are executed one by one. Reversing the order of these two mapping will NOT work.

Examples:

```
> map "tomorrow $1" "add $1 -t 1d"
> tomorrow "Finish user guide"
```

The latter command will be parsed as `add "Finish user guide" -t 1d`, and a new task "Finish user guide" will be added with the deadline to be 1 days later

```
> map "do $1 at $2" "add $1 -t $2"
> do "Laundry" at 4h
```

The latter command will be parsed as `add "Laundry" -t 4h`, and a new task called "Laundry" will be added with the deadline to be 4hours later.

```
> map "ls $0" "ls -f no $0"
> ls
> ls -g cs2103
```

The second command will be parsed as `ls -f no`, and will list out all unfinished tasks. The third command will be parsed as `ls -f no -g cs2103`, and will list out all unfinished cs2103 tasks.

Important: TUI uses `ls` to retrieve tasks. Mapping `ls` to something else will affect behaviour of TUI.

2.5.3 taskManager script

Task manager commands can be saved in a single script file and be executed using run command.

```
$ cat tmscript
ls
map "ls" "ls -f no"
ls
$ ./taskManager
Task Manager V 0.2
exit<enter>to quit.  help<enter>for more instructions
=====================================================================
> run tmscript
1 f Sample task 1.  This also has high priorty
2 Sample task 2.  This has high priority
3 Sample task 3.  This is the latest

2 Sample task 2.  This has high priority
3 Sample task 3.  This is the latest
```

The first 3 tasks are the result of the first ls in the script. The last 2 tasks are the result of the second ls in the script. Because ”ls” is mapped to ”ls -f no”, finished tasks are not shown by the second ls.

TaskManager scripts are plain text files.

2.5.4 startup script

By default, taskManager executes a special script everytime when it is started. This script is `~/.tmrc` on *nix and `%USERPROFILE%\tmrc.txt` on Windows.

This file can be editted to include customized settings.

Examples:

1. To switch to the interactive user interface by default, add this line into tmrc:
   `tui`

2. To save a backup file when taskManager is started:
   `export /tmp/backupTasklist.xml`

3. To show tasks when taskManager is started:
   `ls`

4. To remove finished tasks when taskManager is started:
   `ls -f yes | rm`

5. To run a script with all self-defined mappings when taskManager is started:
   `run /home/myusername/mymappings`

2.5.5 talk to taskManager

For all inputs that cannot be recognized by taskManager as a command, it will be treated as natural language sentense. TaskManager will try its best to recognize it and give correct response.

Example:

```
> what do I do today?
```

All tasks due today will be listed out.

2.6 Compilation and Installation

2.6.1 Microsoft Windows

On Windows, taskManager can be built with Visual Studio in the following steps:

1. Start Visual Studio with "C++ Development Settings".

2. Create a win32 console project.
3. Drag all source files into the solution folder. Files should be automatically categorized into header files and source files.
4. Edit project configuration, under general, set character set to be multibyte characters.
5. Edit project configuration, under linker, add pdcurses.lib to additional libraries and add the folder containing pdcurses.lib to library search directories.
6. Copy pdcurses.dll to %WINDIR%\system32\or the directory your executable will be generated.
7. Build the solution.

To build taskManager with TUI, pdcurses library is needed. It is free and can be downloaded here: `http://sourceforge.net/projects/pdcurses/files/`

Edit the project property to include the pdcurses.lib and in the linker options.

2.6.2 Unix-like Operating Systems

In Unix-like operating systems like GNU Linux and Mac OS X, start a shell, change directory to taskManager's source folder and type:

```
$ make
$ sudo make install
```

"make install" is optional. It just makes taskManager available system wide by copying the executable and man pages to cooresponding directories.

The text UI is built by default, which requires ncurses library. It ships with most Linux distributions and Mac OS. If not, it can be installed with the package manager (`apt-get`, `yum`, `pacman` on various Linux distributions and `port` on Mac).

3 Developer Guide

ahah

4 Milestones and Individual Work