***Experiment No:-11***

**Aim:- Case Study on ld commands for Linux**

***Theory:***

ld is a GNU linker. ld combines a number of objects and archive files, relocates their data and ties up symbol references. Usually the last step in compiling a program is to run ld. ld accepts Linker Command Language files written in a superset of AT&T’s Link Editor Command Language syntax, to provide explicit and total control over the linking process. This version of ld uses the general purpose BFD libraries to operate on object files. This allows ld to read, combine, and write object files in many different format for example, COFF or "a.out". Different formats may be linked together to produce any available kind of object file. Aside from its flexibility, the GNU linker is more helpful than other linkers in providing diagnostic information. Many linkers abandon execution immediately upon encountering an error; whenever possible, ld continues executing, allowing you to identify other errors (or, in some cases, to get an output file in spite of the error). The GNU linker ld is meant to cover a broad range of situations, and to be as compatible as possible with other linkers. As a result, you have many choices to control its behavior.

**ld commands for Linux:**

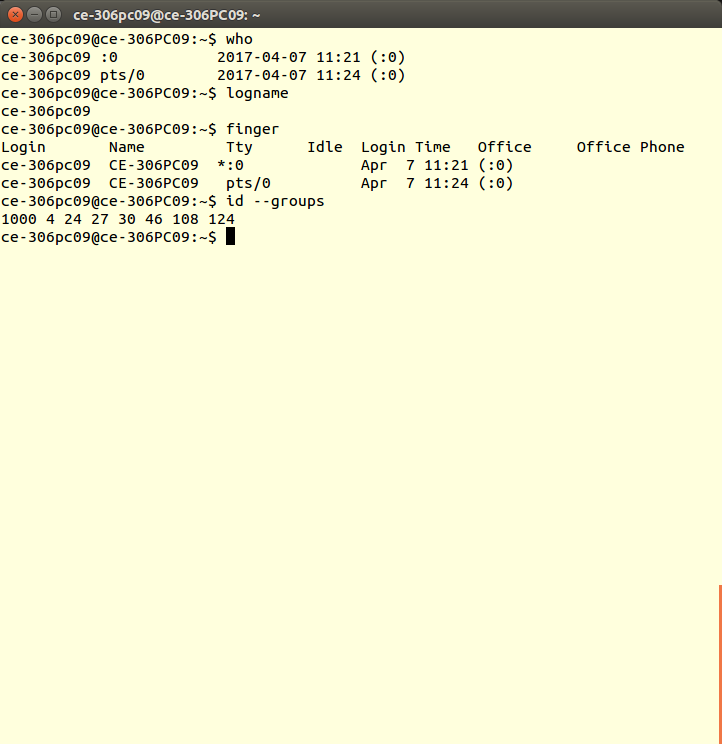
**finger** — List information about a user.

**logname** — Return a user's login name.

**who** — Report which users are logged in to the system.

**-G, --groups**---print all group IDs.

***Execution***

****

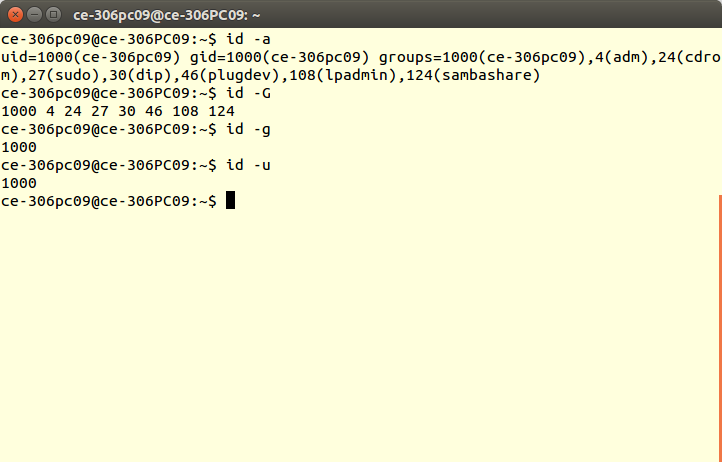
**id -a:** To print user name, UID of all the group to which the user belongs

**id -G:** To print all different group IDs (effective, real and supplementary)

**id -g:** To print the effective group ID

**id -u:** To print the effective user ID

***EXECUTION***

******

***CONCLUSION:*** Hence Linux commands are studied and implemented successfully***.***