Lab: getline in context of STDIN stream and for a file

```
BEGIN {
getline str < "/dev/stdin";                    ← get I/p from keyboard
print "you entered: " str;
}
# Main processing
{ }
```

Lab: Utilise return code of GETLINE

```
BEGIN {
        retval=1;
        while(retval==1) {
        printf "\nInput some data: ";
        retval=getline somevar < "/dev/stdin";
        printf "\nRetval: %d  data entered: [%s]\n",retval, somevar;
        }
}
# main processing
{ }
```

NOTE:
Input some data: <CTRL D pressed which stands for EOF>

Lab: Ask user file name and read from this file

```
BEGIN {
print "Enter Filename to read from: " ;
getline flname < "/dev/stdin";
input_file=flname;
getline str < input_file;
print "File :" FILENAME;
print "Record >> " str;
}
# Main processing
{}
```

Lab: Utilise output of a Unix command using getline()

```
BEGIN {
  FileName="list.txt";
  while( ("sort -r list.txt" | getline record_data) > 0 ) {
          printf("\n %12s", record_data );
       } # while
   close("sort -r list.txt");

} # end: BEGIN
 # main code
{}
```

NOTES: File is created as a result of SORT command.
To close the file, issue CLOSE on the SORT command itself. This is necessary since we do not have a specific file name for the output of the SORT command. file is automatically opened when a GETLINE is done to it.

Try this:

Lab: getline : read from multiple files

INTERNAL

```
# getline function
BEGIN {
print "Filename: " FILENAME;
input_file="test.txt";
getline str < input_file;
print "File :" FILENAME;
print "Record >> " str;
str="";
# read a record from file 'a'
input_file="a";
getline str < input_file;
print "File :" FILENAME ;
print "Record >> " str;
# read a record from current file
getline str;
print "File :" FILENAME ;
print "Record >> " str;
}
# Main processing
{}
```

Lab: Behavior of GETLINE

$ cat bmast 1 B01 100 2 B02 200 3 B03 300 4 B04 400 5 B05 500	$ cat testfile this is line 1 this is line 2 this is line 3 this is line 4

find the difference between the 2 pieces of code below:

PROGRAM #1	**PROGRAM #2**

$ cat awkgetfile
BEGIN { cmd1="cat testfile"; }
# main processing 	
{
record_data=$0;
print record_data;
 	
if((cmd1 | getline)>0) {
print "getfile:" $0;
}
}
END { close(cmd1);}
	$ cat awkgetfile
	BEGIN { cmd1="cat testfile"; file_available=1;}
	# main processing
	{
	record_data=$0;
	print record_data;
	
	if(file_available==1) {
	while((cmd1 | getline)>0) {
	print "getfile:" $0;
	}}
	  if (file_available==1) { file_available=0; close(cmd1);};
	}

EXECUTION awk -f awkgetfile bmast	EXECUTION awk -f awkgetfile bmast
OUTPUT $ awk -f awkgetfile bmast 1 B01 100 getfile:this is line 1 2 B02 200 getfile:this is line 2 3 B03 300 getfile:this is line 3 4 B04 400 getfile:this is line 4 5 B05 500	OUTPUT $ awk -f awkgetfile bmast 1 B01 100 getfile:this is line 1 getfile:this is line 2 getfile:this is line 3 getfile:this is line 4 2 B02 200 3 B03 300 4 B04 400 5 B05 500

Notes: the cmd1 commands holds a valid shell command.

Lab: Re-run the command and start from first record of the buffer (which contains output of cmd)

```
BEGIN { cmd1="cat bmast"; }
# main processing
{
 retval =  cmd1 | getline ;
 if (retval > 0) { print "record:" $0; }
 retval =  cmd1 | getline ;
 if (retval > 0) { print "record:" $0; }
 close (cmd1);   /* re-run the command and start reading fron the first record - re-open the file from memory */
 retval =  cmd1 | getline ;
 if (retval > 0) { print "record:" $0; }
}
```

Lab: ERRNO special variable

```
BEGIN { cmd1="cat testfile";  file_available=1; }
# main processing
 {
 record_data=$0;
 print record_data;

 if(file_available==1) {
 while("1") { if((cmd1 | getline)>0) { print "getfile:" $0; } else { print ERRNO; close(cmd);
break;}  }
if(ERRNO !="") { printf "\n ERROR TEXT: [%s] \n",ERRNO; }
}
   if (file_available==1) { file_available=0; close(cmd1);};
}
```

Notes:

- cmd1 contains the command that will be executed.
- The command executes and produces a set of records. These are stored in pipe.
- One by one, during each call of getline() these records are given to getline
- When no more records are lefft, getline returns a ZERO (EOF). If record is acquired, getline returns a 1.
- in case of error, getline returns a -1. For each call of getline, 1 record is retrieved.