Advance Bash Scripting

Background Process
to run bash script in a background just add & after the script name

```
sleep 60 &
```

this will keep the sleep thread for 60 seconds in a background
along with that it will create **JOB ID** and Process **ID (PID)**
OUTPUT

```
[1] 12345
```

here 1 is the Job ID and 12345 is PID

to get the list of all background process

```
jobs
```

output

```
[1]+ Running Sleep 60 &
```

to see the running process

```
ps aux | grep sleep
```

to send process | script to the foreground

```
fg %1
```

fg- foreground and %1 is job ID
if you will enter only **fg** then it will send the latest process to the foreground

```
fg
```

to send process to background again

```
ctrl+z
bg
```

to KILL the Process

```
kill <PID> or kill 12345 or kill -9 12345
```

to terminate a background job using JOB ID :

```
kill %1
```

here %1 is JOB ID

Running a Process in the Background and Keeping it Alive

nohup- No Hang UP
when we close the terminal process terminated by default, To Prevent This:

```
nohup ./script.sh &
```

- output is saved in nohup.out by default
- use disown

```
./script.sh &
disown
```

this will remove the process from the job table, so it wont be terminated when the session ends

TASK: Running Background Script: create a Script that Generates Random File and Folders , also the script writes the data in the files

1. **STEP:1 use shebang**
2. **STEP:2 write your logic using Bash**
3. **STEP:3 make it executable using chmod + <scriptname>.sh**
4. **STEP:4 run the script with &**

AUTOMATIC USER MANAGEMENT

we can use a script to add multiple users , set passwords and assign permissions

```
#!/bin/bash

USERS=("nikunj" "sonali" "mahesh" "golu")
```

```
for user in "${USERS[@]}"; do

if id "#user" &> /dev/null; then
echo "User $user already exist";
else sudo useradd -m -s /bin/bash "$user"
echo "User $user created."
fi
done
```