DATE:16-06-2022

NAME: SUCHITHRA RAJ M

USN: 1NT19IS162

**BIG DATA LAB**

EXERCISE 3: Use the Hadoop framework to write a custom MapReduce program to perform word count operation on a custom data set.

**package** suchithra162;

**import** java.util.\*;

**import** org.apache.hadoop.fs.Path;

**import** org.apache.hadoop.io.IntWritable;

**import** org.apache.hadoop.io.LongWritable;

**import** org.apache.hadoop.io.Text;

**import** org.apache.hadoop.mapred.FileInputFormat;

**import** org.apache.hadoop.mapred.FileOutputFormat;

**import** org.apache.hadoop.mapred.JobClient;

**import** org.apache.hadoop.mapred.JobConf;

**import** org.apache.hadoop.mapred.MapReduceBase;

**import** org.apache.hadoop.mapred.Mapper;

**import** org.apache.hadoop.mapred.OutputCollector;

**import** org.apache.hadoop.mapred.Reducer;

**import** org.apache.hadoop.mapred.Reporter;

**import** org.apache.hadoop.mapred.TextInputFormat;

**import** org.apache.hadoop.mapred.TextOutputFormat;

**import** java.io.IOException;

**public** **class** WordCount {

**public** **static** **class** Map **extends** MapReduceBase **implements**

Mapper<LongWritable, Text, Text, IntWritable> {

**private** **final** **static** IntWritable ***one*** = **new** IntWritable(1);

**private** Text word = **new** Text();

**public** **void** map(LongWritable key, Text value, OutputCollector<Text, IntWritable>

output, Reporter reporter) **throws** IOException {

String line = value.toString();

StringTokenizer tokenizer = **new** StringTokenizer(line);

**while** (tokenizer.hasMoreTokens()) {

word.set(tokenizer.nextToken());

output.collect(word, ***one***);

}

}

}

**public** **static** **class** Reduce **extends** MapReduceBase **implements**

Reducer<Text, IntWritable, Text, IntWritable> {

**public** **void** reduce(Text key, Iterator<IntWritable> values,

OutputCollector<Text, IntWritable> output, Reporter reporter)

**throws** IOException {

**int** sum = 0;

**while** (values.hasNext()) {

sum += values.next().get();

}

output.collect(key, **new** IntWritable(sum));

}

}

**public** **static** **void** main(String[] args) **throws** IOException{

JobConf conf = **new** JobConf(WordCount.**class**);

conf.setJobName("wordcount");

conf.setOutputKeyClass(Text.**class**);

conf.setOutputValueClass(IntWritable.**class**);

conf.setMapperClass(Map.**class**);

conf.setCombinerClass(Reduce.**class**);

conf.setReducerClass(Reduce.**class**);

conf.setInputFormat(TextInputFormat.**class**);

conf.setOutputFormat(TextOutputFormat.**class**);

FileInputFormat.setInputPaths(conf, **new** Path(args[0]));

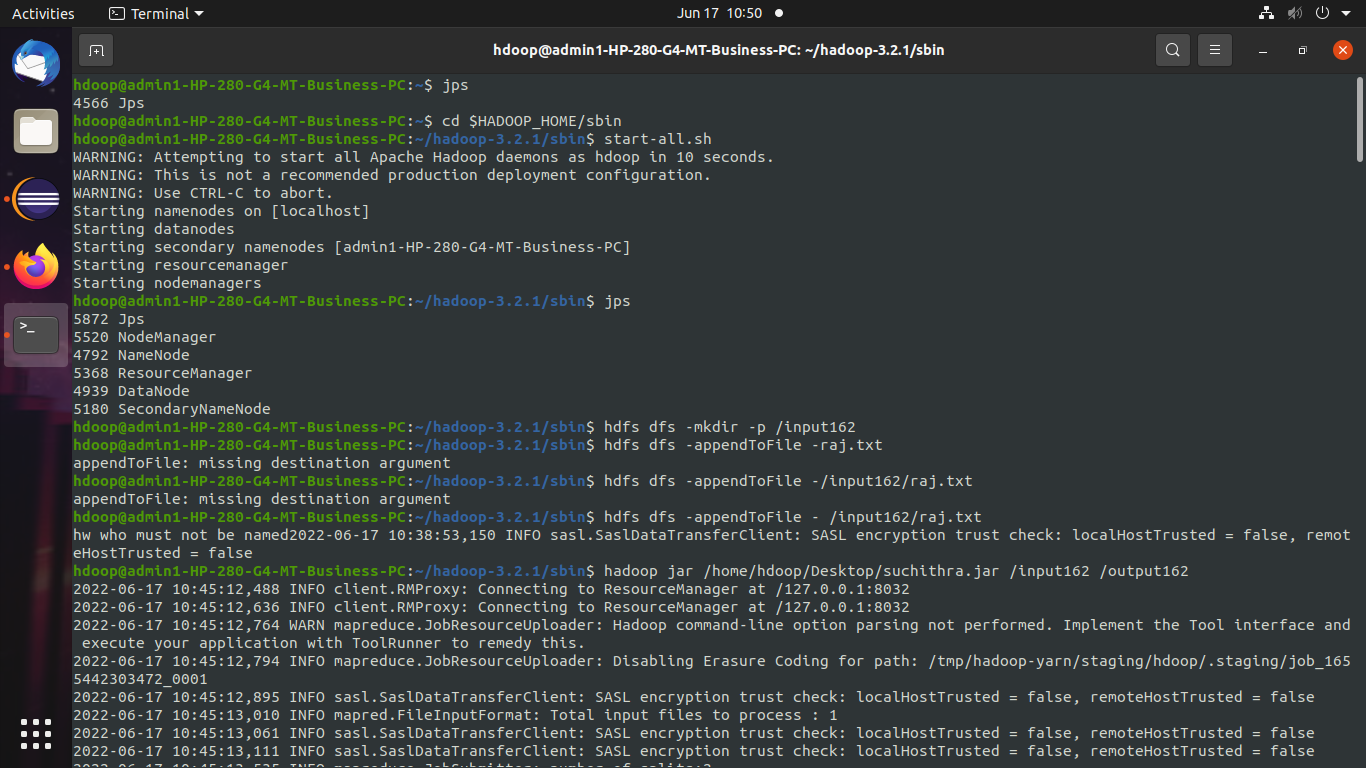
FileOutputFormat.setOutputPath(conf, **new** Path(args[1]));

JobClient.runJob(conf);

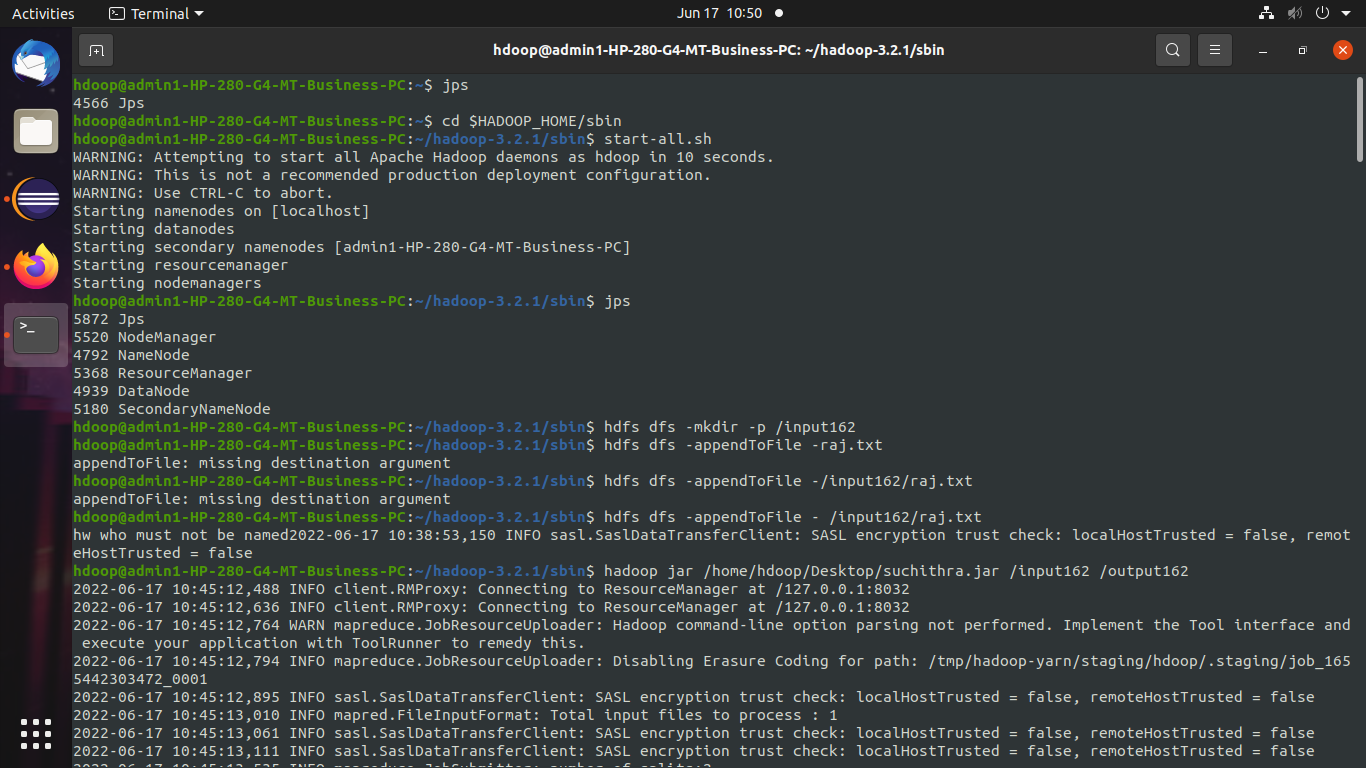
}

}

In terminal, we’ll change the directory to Hadoop-3.2.1/sbin and then we’ll start Hadoop.

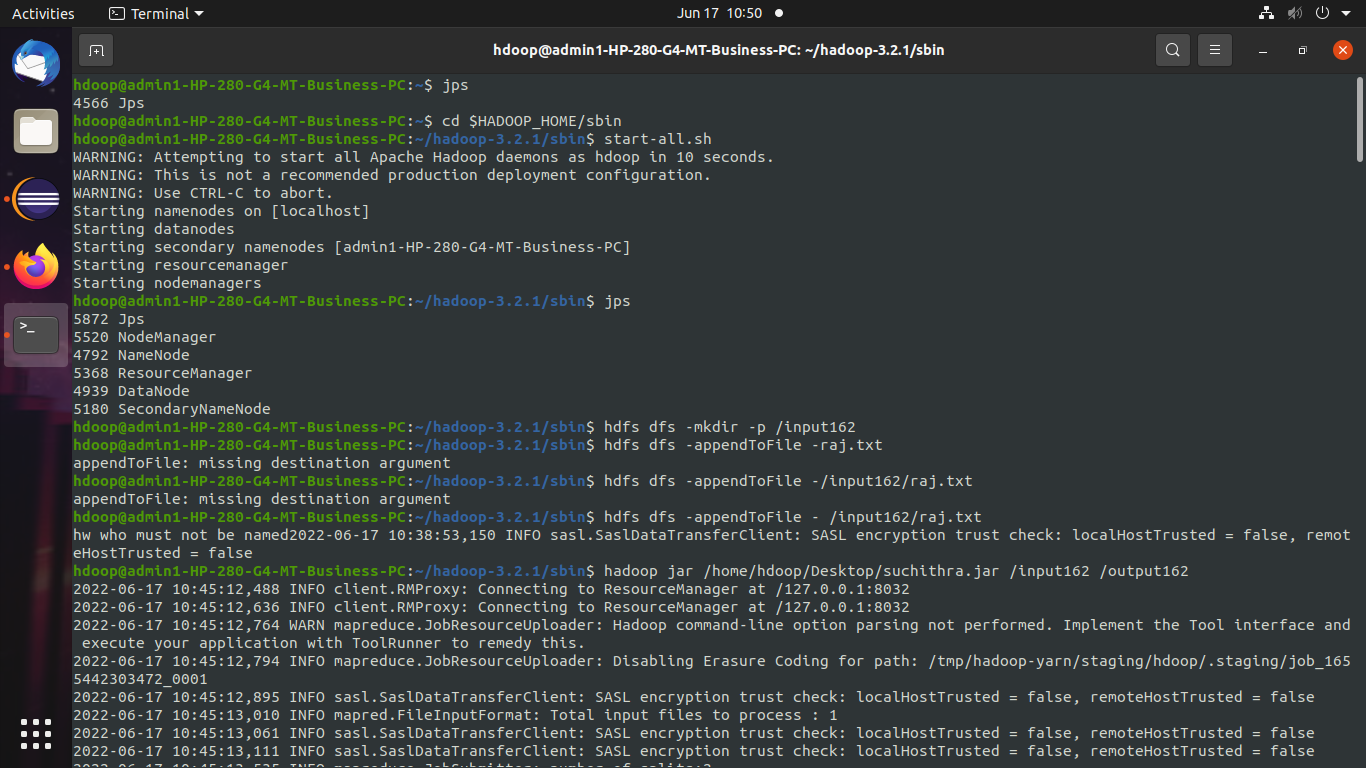


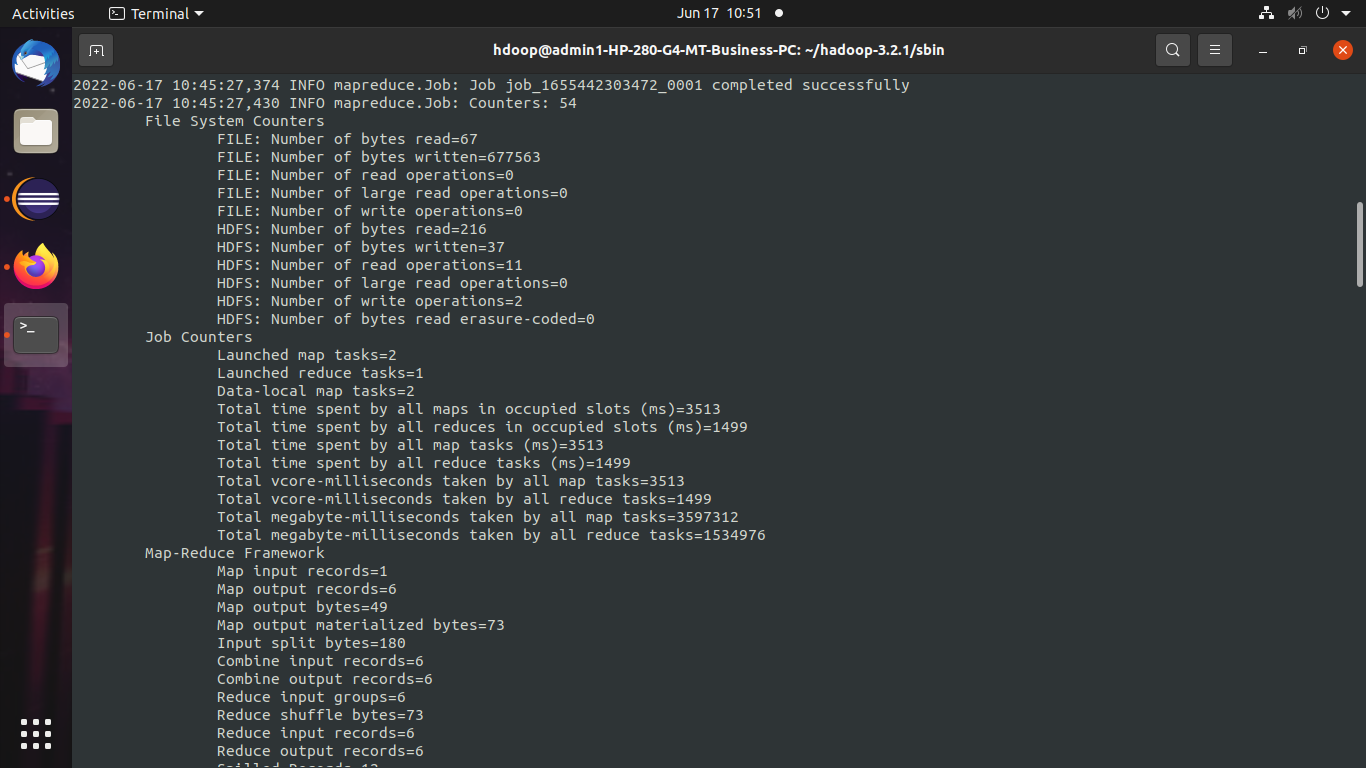
Creating input directory and inside that directory create a file and add text inside the file.



We use hadoop jar /home/hdoop/Desktop/suchithra.jar /input162 /output162

Using the above command, we can print the class path needed to get the Hadoop jar and required libraries.





To see the output, we use hdfs dfs -cat /output162/part\*.

