Set Up Instance	
Create KeyPair (done once)	aws ec2 create-key-pair --key-name MyKeyPair --query 'KeyMaterial' --output text > MyKeyPair.pem chmod 400 ~/MyKeyPair.pem aws ec2 create-key-pair --key-name MyKeyPair2 --query 'KeyMaterial' --output text > MyKeyPair2.pem
Run Instance using KeyPair	aws ec2 run-instances --image-id resolve:ssm:/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2 --instance-type t2.micro --key-name MyKeyPair
Enable SSH	1. Go to EC2 Dashboard → Select Instance → Security → Edit Inbound Rules → Add Rule → "SSH" + "Anywhere IPv4" aws ec2 describe-instances \\ --filters Name=instance-type,Values=t2.micro \\ --query "Reservations[].Instances[].{PublicIpAddress:PublicIpAddress}" ssh -i MyKeyPair.pem ec2-user@<ip_address> To my instance: ssh -i MyKeyPair.pem ec2-user@13.212.185.158
Handling Files	
SCP (to EC2)	scp -i MyKeyPair.pem <file_name> ec2-user@<ip_address>:~/ scp -i MyKeyPair.pem Downloads/ee3801_aws/EE3801_Lab3_Hannah.py ec2-user@18.143.171.189:~/ scp -i MyKeyPair.pem Downloads/ee3801_aws/FAOSTAT_Lab3_Original.csv ec2-user@18.143.171.189:~/
SCP (from EC2)	scp -i MyKeyPair.pem ec2-user<ip_address>:<file_name> <Destination Path> scp -i MyKeyPair.pem ec2-user@18.143.171.189:output_Apples_Production.csv /Users/leeey/Downloads/ee3801_aws
	wget https://repo.continuum.io/miniconda/Miniconda3-latest-Linux-x86_64.sh $ bash Miniconda3-latest-Linux-x86_64.sh

	$ source ~/.bashrc $ conda install pandas
Terminate Instance	
Get InstanceId	aws ec2 describe-instances \\ --filters Name=instance-type,Values=t2.micro \\ --query "Reservations[].Instances[].{InstanceId:InstanceId}" aws ec2 describe-instances \\ --filters Name=instance-type,Values=t2.micro \\ --query "Reservations[].Instances[].{PublicDnsName:PublicDnsName, LaunchTime:LaunchTime}"
Terminate	aws ec2 terminate-instances --instance-ids <ids-here> aws ec2 terminate-instances --instance-ids i-012158314717cb854
Starting a cluster in an EC2 Instance	
Configuration Install Amazon ParallelCluster	 https://docs.aws.amazon.com/parallelcluster/latest/ug/install-v3-pip.html (if node.js not found, use an older version, EC2 OS is not yet compatible for newer versions) 'nvm install --lts=Gallium'

My instance:
EC2 Instance Created:
"InstanceId": "i-02efcdfd7ddbfec5d"
Public IPv4 Address: 13.212.185.158

If missing configurations:
https://repost.aws/knowledge-center/s3-locate-credentials-error

Cluster commands

Creating a Cluster from a Snapshot	
`Edit config file in terminal`	`nano ~/cluster-config.yaml` Change snapshot id to the latest version

Create the Cluster
+
SSH in
	`pcluster create-cluster -c ~/cluster-config.yaml -n MyCluster01`
	
	`pcluster ssh -i ~/MyKeyPair.pem -n MyCluster01`
	
	`[if pcluster not found:`
	`source ~/.bash_profile]`

Describe cluster	`pcluster describe-cluster -n MyCluster01`
Activating Miniconda	

Initialise
+
Reload bash
	`(in /data directory)`
	`miniconda3/bin/conda init`
	
	`source ~/.bashrc`

Conda activate	`conda activate env1`

Copy aws credentials
+
Check that credentials are correct
	`cp -r /data/aws ~/.aws`
	
	`aws sns publish --topic-arn arn:aws:sns:ap-southeast-1:532875939626:awsnotify --message "ClusterTest"`
	
	Copying credentials and KeyPair from EC2 instance to Cluster:
	`scp -rp -i ~/MyKeyPair.pem ~/.aws ~/MyKeyPair.pem ec2-user@52.221.212.153:~/`

AWS Configure	aws configure Default Region Name: ap-southeast-1 Default Output Format: json AWS Root user key: - AKIAXYEPBR4VC23BLPZZ - 0R7f5HWa2fM3au8obEVXfu0IQt29TsRzStf3N/U8
Snapshots	

Take a snapshot (in own terminal not SSH)	`update_snapshot.sh data 2 MyCluster01`
	
	`OR`

	`chmod a+x ~/Downloads/ee3801_aws/PyHipp/update_snapshot.sh` [optional] `~/Downloads/ee3801_aws/PyHipp/update_snapshot.sh data 2 MyCluster01`
Check that Snapshot was created	`aws ec2 describe-snapshots --owner-ids self  --query 'Snapshots[]'`
Delete & Shutdown Cluster	
	`pcluster delete-cluster -n MyCluster01`
Modify active cluster configuration	
Make edits to cluster-config.yaml file first	
Stop compute node (if no jobs running)	`pcluster update-compute-fleet --status STOP_REQUESTED --region ap-southeast-1 --cluster-name MyCluster01`

Update cluster (if u changed cluster-config.yaml)	`pcluster update-cluster --cluster-configuration ~/cluster-config.yaml --cluster-name MyCluster01`
	
	`If update failed, check status of compute node:`
	`pcluster describe-cluster --region ap-southeast-1 --cluster-name MyCluster01`

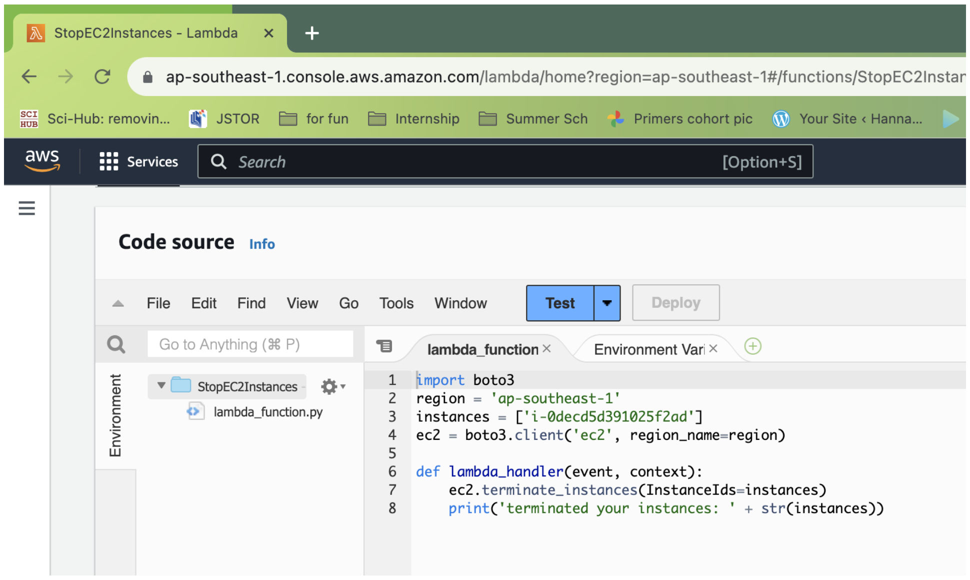
Start compute nodes (if they didnt restart automatically after update)	`pcluster update-compute-fleet --status START_REQUESTED --region ap-southeast-1 --cluster-name MyCluster01`
EC2 and Cluster	
SCP from EC2 to Cluster	`scp -i ~/MyKeyPair.pem ~/MyKeyPair.pem ec2-user@13.229.184.190:/data`
Using compute nodes	

Log in to compute node	`srun --pty /bin/bash`
	
	`If srun or other slurm commands not found, then exit and re ssh into the cluster`

AWS Lambda (automatically delete head node)	
	- Create EC2 Instance first & take note of InstanceId https://ee3801.github.io/Lab6/part-b/6.html

(Part 6)

- Make amendments to this lambda function that terminates a cluster when a snapshot has been completed by editing the instances with the instance id of the cluster


StopEC2Instances - Lambda
ap-southeast-1.console.aws.amazon.com/lambda/home?region=ap-southeast-1#/functions/StopEC2Instar
Code source Info
File Edit Find View Go Tools Window Test Deploy
lambda_function
Environment Vari
StopEC2Instances
lambda_function.py

```python
import boto3
region = 'ap-southeast-1'
instances = ['i-0decd5d391025f2ad']
ec2 = boto3.client('ec2', region_name=region)

def lambda_handler(event, context):
    ec2.terminate_instances(InstanceIds=instances)
    print('terminated your instances: ' + str(instances))
```

Github personal access token: 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

Github Things

Getting PyHipp
 	
Installation
	`cd /data/src`
	`git clone https://github.com/leeey03/PyHipp.git`
	
	`cd PyHipp; pip install -r requirements.txt; pip install -e .; cd /data`

Setting up Github	
Set up origin	`git remote set-url origin https://github.com/leeey03/PyHipp.git`
Set up upstream repo	`git remote add upstream https://github.com/shihchengyen/PyHipp.git`
Check that upstream is set properly	`git remote -v`
Get the repo	Git pull
Merging with Upstream	
`Deconflict Files`	Can rename duplicate files mv {OG_filename} {new_filename}

Merging	`git fetch upstream`
	
	`git checkout main`
	
	`git merge upstream/main`

Undoing merge to resolve conflicts	`git reset --merge`
Resolving conflicts (ghetto style)	https://opensource.com/article/23/4/resolve-git-merge-conflicts
Add file to github repo	

	`git config --global core.editor "nano"`
	
	`git add <filename>`
	
	`git commit -m “message here”`
	
	`git push`

Unix things:
https://nusit.nus.edu.sg/wp-content/uploads/2019/09/unixcom.pdf

EE3801 EXAM CHECKLIST

1. Spyder + libraries have been pip installed and are functioning
    a. Spyder commands needed (lab 4 + lab 8b)
2. Github is updated ! can pull
3. Bash Cheatsheet

PyHipp

1. RPLParallel (for both session01 and sessioneye)
2. RPLSplit to create a RPLRaw object for each of the 110 channels (for both session01 and sessioneye)
3. RPLLFP (which needs the RPLRaw object) for each of the 110 channels (for both session01 and sessioneye)
4. RPLHighPass (which needs the RPLRaw object) for each of the 110 channels (for both session01 and sessioneye)
5. Spike sorting (which needs the RPLHighPass objects for both session01 and sessioneye) for each of the 110 channels
6. Unity (needs RPLParallel object)
7. EDFSplit to create Eyelink objects (needs RPLParallel, and Unity if available) (for both session01 and sessioneye)
8. Aligning_objects (needs RPLParallel, Unity, and Eyelink objects)
9. Raycasting (needs Unity and Eyelink objects)

10. Creating cumulative objects
    a. uyall (Needs Unity Objects)
    b. wfall (Needs results of spike sorting)
    c. FreqSpectrum (needs RPLLFP and RPLHighPass objects)

Processing Ripple Data	

Create RPLParallel object	`Processes Ripple (.nev) files sent by Unity:`
	`- Creates rplparallel*.hkl files ie. RPLParallel Objects`
	
	`cd /data/picasso/20181105/session01`
	`pyh.RPLParallel(saveLevel=1)`

	`Can be used to process the eye fixation session:`
	`cd ../sessioneye`
	`pyh.RPLParallel(saveLevel=1)`

Create RPLRaw Object	Process raw neural data within .ns5 files (found in session and sessioneye) for a single channel009 of 110 channels:
	
	`cd ../session01`
	`pyh.RPLSplit(channel=[9])`
	
	Outcome:
	- Creates an arrayXX folder with subfolder channelXXX and a .hkl file within named rplraw_*.hkl

Create low-pass filtered signals (using RPLLFP function)	Loads rplraw_*.hkl file to create rpllfp_*.hkl file with low pass signals:
	- Run within the channel directory
	
	`cd array01/channel009`
	`pyh.RPLLFP(saveLevel=1)`

Create high-pass filtered signals (using `RPLHighPass` function)	Loads rplraw_*.hkl file to create rplhighpass_*.hkl file with high pass signals:
	- Run within the channel directory
	
	`cd array01/channel009`
	`pyh.RPLHighPass(saveLevel=1)`

Spike sorting	Done in channel directory and creates mountain files (.mda) within cell directories:
	
	`cmd='import PyHipp as pyh; from PyHipp import mountain_batch; mountain_batch.mountain_batch(); from PyHipp import export_mountain_cells; export_mountain_cells.export_mountain_cells();')`
	
	Counting spike sorting output files (one per channel):
	`find mountains -name "firings.mda" | wc -l`
	
	Scripts:
	sort-slurm.sh

Process Unity Data	
	In session01 folder, uses rplparallel_*.hkl file previously created to speed up processes: - If RPLParallel Object not present, will compute needed info from the raw data files - Creates unity_*.hkl file `pyh.Unity(saveLevel=1)`

Process Eyelink Data	
	Process eye-tracking files (.edf) in `20181105 folder:` - `Creates eyelink_*.hkl file` `pyh.EDFSplit()` - Ignore “Serialized Warning/DataFrame” messages
Align Ripple, Unity and Eyelink Data	
Align	In session01: `pyh.aligning_objects()`

Raycast	In session01:
	- Creates a log file eg, VirtualMazeBatchLog.txt
	`pyh.raycast(1)`
	
	`Check in on progress with tail -f filename.txt`

Pipelining

Performing entire analysis on selected channels	See mypipeline-slurm.sh (to be submitted in day directory eg. 20181105) - DPT.objects.processDirs - Switches to correct directory for processing immediately - When used with level=’channel’, cmd’<python script>’ - Finds the appropriate channel directory before running the <python script> specified - DO NOT USE dirs= argument
Split into RPLRaw and RPLParallel processing pipelines	RPLParallel (highlighted in blue): myrplparallel-slurm.sh RPLRaw (highlighted in purple): Myrplsplit-slurm.sh - Do not use DPT dirs or exclude for this Note that the above 2 files are only working on 8 selected channels: PARALLELISING: - rs1-slurm.sh files are used to parallelise working on all 110 channels - Uses DPT.objects.processDirs(dirs=['sessioneye/array01','session01/array01']) to specify the directory to run a command or create an object - NOTE: if using cmd=’python script’, can use dirs and level

	- BUT: if using objtype=pyh.RPLXXX, must set dirs=None, can specify channel=[*range(33,97)]]
	- Note that spike sorting script will combine highpass files in both sessioneye and session01 directories. We do a os.chdir('session01/array01') before running spike sorting to ensure that only channels in the first array are processed
	
	- NOTE: split of channels:
	`channel=[*range(1,33)]`
	`channel=[*range(33,65)]`
	`channel=[*range(65,97)]`
	`channel=[*range(97,125)]`
	- `NOTE that the slurm scripts use 5 CPUs due to the limited memory (40GB) of each compute node`
	
	- Use pipe2.sh to submit all the parallelised jobs tgt instead of entering sbatch 5 times
	
	PARALLELISING + Just in Time JOB SUBMISSION
	- rse-slurm.sh script processes the sessioneye files:
	- rs1a-slurm.sh scripts uses SkipHPC=False, SkipLFP=False, SkipHighPass=False, SkipSort=False to process session01 files:
	  - Look in `HPCScriptsDir for rpllfp-slurm.sh and submit it from channel directory the moment rplraw file is created`
	  - `Does the same for rplhighpass-sort-slurm.sh to create rplhighpass files`
	- pipe2a.sh makes the rs*a-slurm.sh scripts dependent on rse-slurm.sh completion

Using envlist	In 20181101:
	`pip install filelock`
	
	Create envs:
	`x=0; while [ $x -le 63 ]; do echo $x; conda create --name cenv$x --clone env1 --copy; (( x++ )); done; aws sns publish --topic-arn arn:aws:sns:ap-southeast-1:123456789012:awsnotify --message "CondaCreateComplete"`
	
	Check envs:
	`conda env list`
	
	Run envlist.py to create files envlist.hkl and envlist.hkl.lock:
	`/data/src/PyHipp/envlist.py cenv 64`
	
	In slurm scripts, add the following before and after python commands:

	`/data/miniconda3/bin/conda init`
	`source ~/.bashrc`
	``envarg=`/data/src/PyHipp/envlist.py` ``
	`conda activate $envarg`
	
	`PYTHON STUFF`
	
	`conda deactivate`
	`/data/src/PyHipp/envlist.py $envarg`

Checkfiles Checkfiles2.sh removefiles.sh	Counts the number of output files from the above processes (run in day directory eg `20181105` Does the above and also outputs the start and end time of each process Deletes all files created while processing
Check and Resubmit jobs	

Checking for channels without rplhighpass files and resubmitting jobs for those channels	In 20181101 directory:
	
	Create a list of all channels (not in eye or mountain directories):
	`find . -name "channel*" | grep -v -e eye -e mountain | sort > chs.txt`
	
	Create a list with all channels with rplhighpass file:
	`find . -name "rplhighpass*hkl" | grep -v -e eye | sort | cut -d "/" -f 1-4 > hps.txt`
	- cut command prints selected fields separated by delimiter -d "/"
	- -f specifies to print
	comm -23 chs.txt hps.txt

Comparing when paths look different
 	
(finding channels where firings.mda files were not created)
	# js want to keep the channel output, so 3rd column only
	find . -name "firings.mda" | grep -v -e eye | sort | cut -d "/" -f 3 > spike.txt
	
	remake the chs.txt with only the channel names
	find . -name "channel*" | grep -v -e eye -e mountain | sort | cut -d "/" -f 4 > chs2.txt
	
	#compare the two new files
	comm -23 chs2.txt spike.txt
	
	# find the appropriate path to each missing channel??? in chs.txt.
	chs_file="chs.txt"
	
	# Check if the files exist
	if [! -f "$chs_file"]; then
	        echo "File chs.txt does not exist."

	```
	        exit 1
	fi

	# Read lines from chs2.txt into an array
	IFS=$'\n' read -d '' -r -a chs_lines < "$chs_file"

	# Generate the list of lines from the output of comm -23
	search_lines=$(comm -23 <(sort chs2.txt) <(sort spike.txt))

	# Loop through each line in chs.txt
	for chs_line in "${chs_lines[@]}"; do
	  # Loop through each line to search for
	        for spike_line in $search_lines; do
	         # Check if the chs_line contains the spike_line
	        if [[ $chs_line == *"$spike_line"* ]]; then
	        echo "$chs_line"
	        break
	        fi
	        done
	done
	```
	cwd=\`pwd\`; for i in \`bash test_compare.sh\`; do echo $i; cd $i; sbatch /data/src/PyHipp/rplhighpass-sort-slurm.sh; cd $cwd; done

Slurm commands (re-ssh in if got error)	
Submitting jobs	sbatch <slurm.sh>

Submitting jobs with dependencies	afterok makes sure that the jobs are run successfully before running the dependency:
	`sbatch --dependency=afterok:<JOBID>:<JOBID>:<JOBID> /data/src/PyHipp/<dependentJOB>.sh`
	
	afterany will run dependency jobs whether jobs have succeeed or failed
	`sbatch --dependency=afterany:<JOBID>:<JOBID>:<JOBID> /data/src/PyHipp/<dependentJOB>.sh`

Checking on jobs	squeue
Cancel jobs	scancel <jobID> Cancel a range of jobs: scancel {2..7} Cancel all jobs: `scancel --user=ec2-user`

Creating Cumulative Files & Plotting in Spyder

Creating Cumulative Objects	
Appending all Unity objects together	Log in to compute node: `srun --pty /bin/bash` In picasso folder: `cd /data/picasso` `ipython` `In [ ]: import PyHipp as pyh` `In [ ]: import DataProcessingTools as DPT` `In [ ]: uyall = DPT.objects.processDirs(dirs=None, objtype=pyh.Unity)` `In [ ]: uyall.save()`
Creating Waveform objects (display spike sorting results)	`In [ ]: wfall = DPT.objects.processDirs(dirs=None, exclude=['*eye*', '*mountains*'], objtype=pyh.Waveform, saveLevel=1)` `In [ ]: wfall.save()`
SCP objects to local (storing files in Downloads/ee3801_aws)	In local terminal: cd Downloads/ee3801_aws scp -i ~/MyKeyPair.pem -p "ec2-user@13.250.125.132:/data/picasso/unity*.hkl" picasso/ scp -i ~/MyKeyPair.pem -p "ec2-user@13.250.125.132:/data/picasso/waveform*.hkl" picasso/
Create FreqSpectrum Objects - If only doing in a single array, rmb to use the exclude=[*array XX*	Requires the rpllfp and rplhighpass objects: `See freq-slurm.sh` `# LOAD MODULES, INSERT CODE, AND RUN YOUR PROGRAMS HERE` `python -u -c "import PyHipp as pyh; \` `import time; \` `pyh.FreqSpectrum(saveLevel=1); \` `pyh.FreqSpectrum(loadHighPass=True, pointsPerWindow=3000, saveLevel=1);` `print(time.localtime());"` Create the cumulative object of the FreqSpectrum objects: See fsall-slurm.sh (must only be run after freq-slurm.sh) # LOAD MODULES, INSERT CODE, AND RUN YOUR PROGRAMS HERE python -u -c "import PyHipp as pyh; \ import time; \ import DataProcessingTools as DPT; \ lfall = DPT.objects.processDirs(dirs=None, exclude=['*eye*', '*mountains*'], objtype=pyh.FreqSpectrum, saveLevel=1); \ lfall.save(); \

	hfall = DPT.objects.processDirs(dirs=None, exclude=['*eye*', '*mountains*'], objtype=pyh.FreqSpectrum, loadHighPass=True, pointsPerWindow=3000, saveLevel=1); \\ hfall.save(); print(time.localtime());" ALT: see consol_fsjobs.sh: - Same as fsall-slurm but has a dependency on freq-slurm.sh being run first

SCP objects to local	Cd Downloads/ee3801_aws
	`scp -i ~/MyKeyPair.pem -p "ec2-user@13.250.125.132:/data/picasso/freqspectrum_9c80.hkl" picasso/`
	`scp -i ~/MyKeyPair.pem -p "ec2-user@13.250.125.132:/data/picasso/freqspectrum_660e.hkl" picasso/`

Plotting in Spyder (https://ee3801.github.io/Lab8/part-b/4.html)	

Unity Plotting
- Can right click → PlotType → Routes
- Rightclick → PlotType → Proportion of Trials
	`In[ ]: import PyHipp as pyh`
	`In[ ]: import PanGUI`
	`In[ ]: cd ~/Documents/picasso`
	`In[ ]: uy = pyh.Unity(loadFrom='unity_71bf.hkl')`
	`In[ ]: puy = PanGUI.create_window(uy)`
	
	If error is raised change PanGUI/PanGUI/main.py line 173 to remove exclusive=True argument:
	`ag = QtWidgets.QActionGroup(self`~~`, exclusive=True`~~`)`

Waveform plotting
- PlotType → Array
	`In[ ]: wf = pyh.Waveform(loadFrom='waveform_ed79.hkl')`
	`In[ ]: pwf = PanGUI.create_window(wf)`

FreqSpectrum Object plottings	`lf = pyh.FreqSpectrum(loadFrom='freqspectrum_9c80.hkl')`
	`plf = PanGUI.create_window(lf)`
	
	`hf = pyh.FreqSpectrum(loadFrom='freqspectrum_9c80.hkl')`
	`phf = PanGUI.create_window(hf)`

Bash

Print last few lines of a file

```
tail -f VirtualMazeBatchLog.txt
```

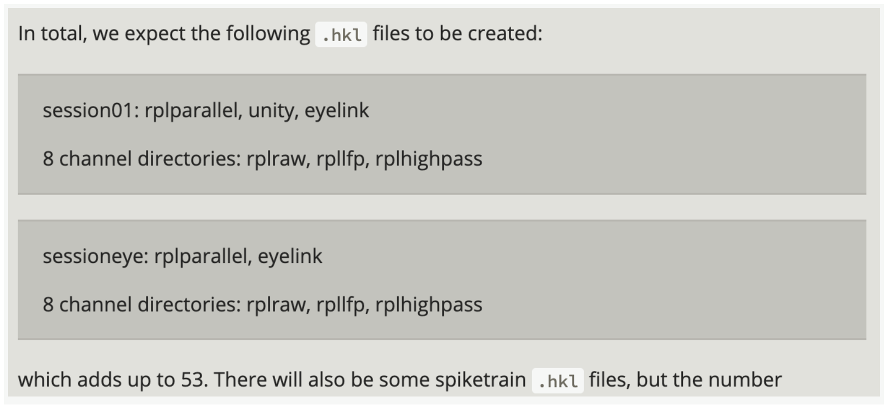
Finding shit

```
Looks for running processes with string “ipython”:
ps -ef | grep ipython

ps
   -  Lists running processes
grep
   -  search function
   -  -v -e <name> <name>
          -  Return files that do not contain the above
             <name>s

Counting all.hkl files (that are not spiketrain files):

find . -name "*.hkl" | grep -v -e spiketrain -e
mountains | wc -l
   -  Grep function returns path of files eg.
      “./session01/eyelink_24d5.hkl”
```



In total, we expect the following `.hkl` files to be created:

session01: rplparallel, unity, eyelink

8 channel directories: rplraw, rpllfp, rplhighpass

sessioneye: rplparallel, eyelink

8 channel directories: rplraw, rpllfp, rplhighpass

which adds up to 53. There will also be some spiketrain `.hkl` files, but the number

```
Finding the running processes:
squeue | grep R
Counting number of running processes:
squeue | grep R | wc -l


Finding output times of specific files, sorted in
alphabetical order:
session01 -name "rplhighpass-sort*out" -or -name
"rpllfp*out" | xargs tail -n 1 | sort

Putting output into a txt file:
find . -name "channel*" | grep -v -e eye -e mountain |
sort > chs.txt
```

Check file sizes

```
find . -name "*.hkl" | grep -v -e spiketrain -e
```

	`mountains | xargs ls -hl` - Pipe to xargs ls -hl function. Xargs takes output of the previous grep function and appends it to the end of ls function -
Kill processes	kill -9 <process_id> - Process_id can be found using ps
copy	cp <OG_FILE> <COPIED_FILE>

While loop	Example that creates clones of environments:
	`(env1) [ec2-user@ip-10-0-5-43 20181105] $ x=0; while [ $x -le 63 ]; do echo $x; conda create --name cenv$x --clone env1 --copy; (( x++ )); done; aws sns publish --topic-arn arn:aws:sns:ap-southeast-1:123456789012:awsnotify --message "CondaCreateComplete"`
	
	Example that finds all channels in a directory and submits jobs within them:
	
	`#!/bin/bash`
	`` cwd = `pwd` ``
	`` for i in `find . -name "channel*" | sort` ``
	`do`
	`    echo $i`
	`    cd $i`
	`    sbatch /data/src/PyHipp/XXXX-slurm.sh`
	`    cd $cwd`
	`Done`
	
	If you need to exclude certain channels, (to get only array02):
	
	`` Use `find . -name “channel*” | grep -v -e mountain -e eye -e array01 -e array03 -e array04 | sort` ``
	
	`Otherwise: js cd into that directory to run what u need`