University of Zurich$^{\text{UZH}}$

High Performance Computing
Lecture 4

Douglas Potter

Cesare Cozza

Mark Eberlein

The Shell Prompt?

```
dhcp-94-215:~$ ssh ela
Last login: Mon May 30 14:16:07 2022 from dhcp-94-190.physik.uzh.ch

  ========================================================================
               IMPORTANT NOTICE FOR USERS of CSCS facilities
      Documentation: CSCS User Portal - https://user.cscs.ch
      Request support: https://support.cscs.ch
  ========================================================================



[dpotter@ela1 ~]$ ssh eiger
Last login: Tue Oct 11 09:44:46 2022 from 148.187.1.10

========================================================================
             IMPORTANT NOTICE FOR USERS of Alps (EIGER)

  Documentation: Alps User Guide - https://confluence.cscs.ch/x/_gD0E
  Request support: Service Desk at https://support.cscs.ch
========================================================================


[eiger][dpotter@nid001080 ~]$
```

What is this?

“C”

Scripting

RegEx

- C Crash Course (memory & pointers)
- Benefits of Scripting / Why Script?
  - Makes repetitive tasks faster
  - Reduces errors
  - Orchestration
- Focus on BASH scripting
  - Many other good tools exist (e.g., Python)
  - BASH works well for program Orchestration
  - BASH is omnipresent (on every Linux System)
  - Usually BASH is the default shell
  - You will encounter BASH in HPC
- Regular Expressions

C Data Types (**signed** or **unsigned**)

Type	Size in bits (bytes)	Description & Maximum Range (may be larger)
char	8 (1)	Smallest addressable unit on the machine [-127,+127] or [0,255]
short	16 (2)	At least [-32767,+32767] or [0,65535] (signed or unsigned)
int	16 (2) – 32 on daint	At least short, but may be long. Natural size of the machine
long	32 (4) – 64 on daint	[−2,147,483,647, +2,147,483,647] or [0, 4,294,967,295] (or larger)
long long	64 (8)	[-$2^{63}-1$, +$2^{63}-1$] or [0, $2^{64}-1$]
float	32 (4)	IEEE 754 single-precision floating-point format
double	64 (8)	IEEE 754 double-precision floating-point format
long double	80, 96, 128 (10–16)	IEEE 754 quadruple-precision or other supported format

Fixed width integer types: #include <stdint.h>

Type	Size in bits (bytes)	Description & Maximum Range (may be larger)
int8_t	8 (1)	[-128,+127]
uint8_t	8 (1)	[0,255]
int16_t	16 (2)	[-32768,+32767]
uint16_t	16 (2)	[0,65535]
int32_t	32 (4)	[−2,147,483,648, +2,147,483,647]
uint32_t	32 (4)	[0, 4,294,967,295]
int64_t	64 (8)	[-2^{63}, $+2^{63} - 1$]
uint64_t	64 (8)	[0, $2^{64} - 1$]

Better fixed width integer types?

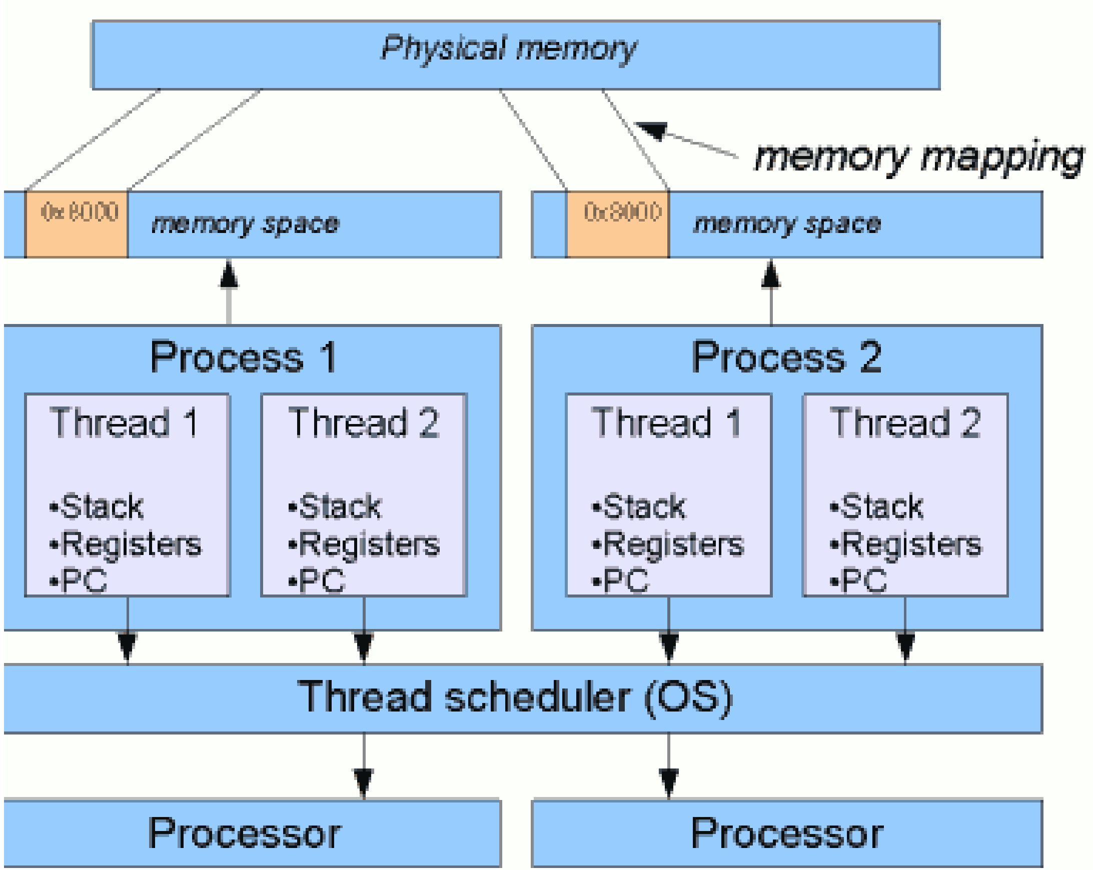
Type	Size in bits (bytes)	Description & Maximum Range (may be larger)
int_least8_t	>= 8 (1)	[-128,+127]
uint_least8_t	>= 8 (1)	[0,255]
int_least16_t	>= 16 (2)	[-32768,+32767]
uint_least16_t	>= 16 (2)	[0,65535]
int_least32_t	>= 32 (4)	[−2,147,483,648, +2,147,483,647]
uint_least32_t	>= 32 (4)	[0, 4,294,967,295]
int_least64_t	>= 64 (8)	[-2^{63}, $+2^{63} - 1$]
uint_least64_t	>= 64 (8)	[0, $2^{64} - 1$]

Even better fixed width integer types?

Type	Size in bits (bytes)	Description & Maximum Range (may be larger)
int_fast8_t	>= 8 (1)	[-128,+127]
uint_fast8_t	>= 8 (1)	[0,255]
int_fast16_t	>= 16 (2)	[-32768,+32767]
uint_fast16_t	>= 16 (2)	[0,65535]
int_fast32_t	>= 32 (4)	[−2,147,483,648, +2,147,483,647]
uint_fast32_t	>= 32 (4)	[0, 4,294,967,295]
int_fast64_t	>= 64 (8)	[-2^{63}, $+2^{63}-1$]
uint_fast64_t	>= 64 (8)	[0, $2^{64}-1$]

Let’s Talk Memory

- All processes have memory
- All variables are in memory
- Think of memory as an array
  - Starts at 0
  - Goes to $2^{64}-1$
  - Not all indexes are valid (most not)
  - Each element is a “byte”
- Bigger types take more bytes and have consecutive addresses.



Neil Coffey

Basic Types

```
int i;
float f;
char c;
i = 12;
f = 23.88;
c = ’X’;
```

Compound Type (structures)

```
struct person {
    float height;
    int age;
};

struct person bob;
bob.height = 1.91;
bob.age = 24;
```

Pointers

C Code	Bytes	What is it?
int i;	4	An integer somewhere in memory. Compiler keeps track of where.
int *p;	8	A pointer to an integer. Contains the address of the integer.
&i	8	Pointer to (address of) the integer ”i”.
p	8	Address of an integer.
*p	4	What’s at the address “p” (an integer)
int **q	8	A pointer to a pointer to an integer. e.g.: q = &p
*q	8	A pointer to an integer.
**q	4	An integer.

Structure Pointers

```
struct person {
    float height;
    int age;
};

struct person bob;

bob.height = 1.91;
bob.age = 24;
```

```
struct person grace;
Struct person *p = &grace;

(*p).age = 40;

p->height = 1.79;
```

Dynamic Memory (allocation)

C Code	What it means
float f;	A single precision floating point number somewhere in memory
float *p = &f;	A variable that has the address of “f” (points to “f”)
float *q;	Another “pointer”. **The value is undefined! Don’t use it (yet).**
q = malloc(4);	Allocates four bytes and now “q” has the address. **Don’t do this.**
q = malloc(sizeof(float));	Same as before, but the compiler fills in the number of bytes needed.
q = malloc(100*sizeof(float));	Allocate 400 bytes (100 x 4) to store 100 float values.
free(q);	Free memory that “q” points to.

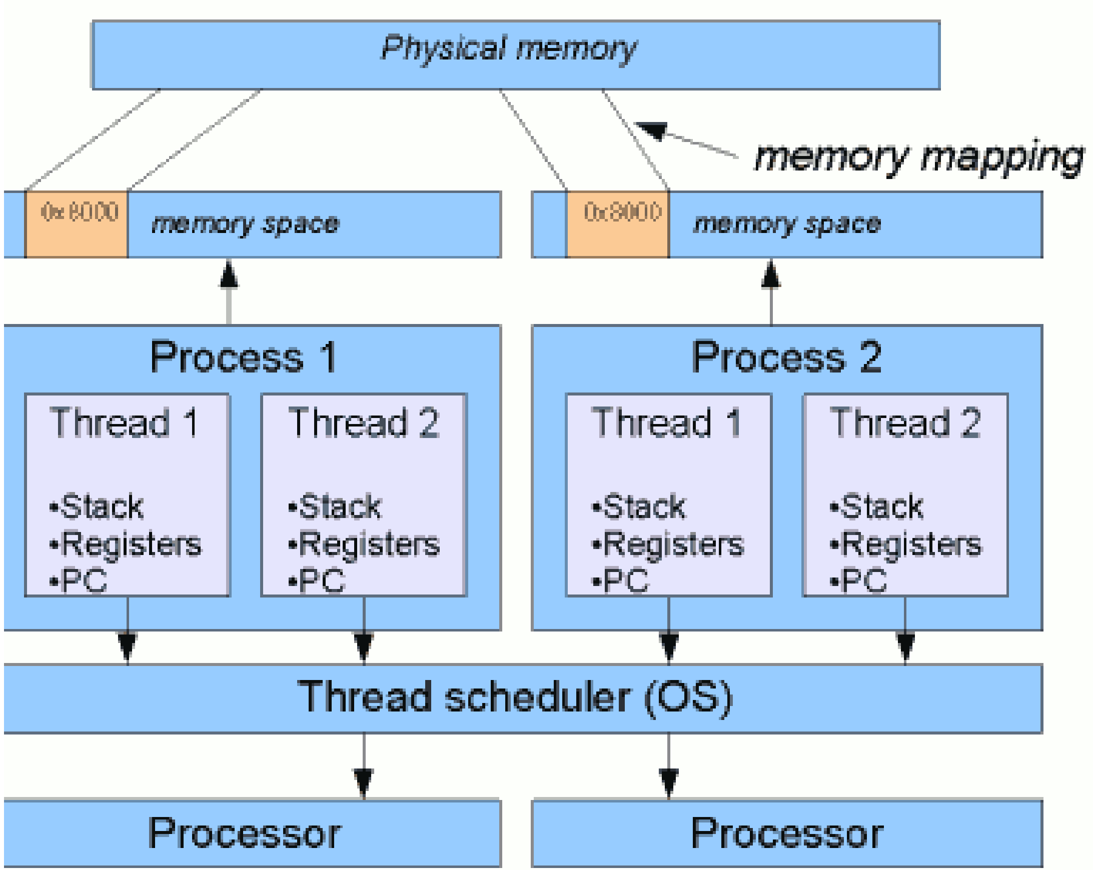
Pointer arithmetic

C Code	What it means
float *p = malloc(100*sizeof(float));	Allocate memory for 100 floating point numbers
*p = 3.14;	Set the “float” at p to 3.14. So set the **first** float of 100.
p[0] = 8.1;	Set the first float at p to 8.1.
p[1] = -4.0;	Set the second float to -4
*(p+1) = -4.0;	Same as above. Add “1 float” to the address “p” and dereference it.
p[-1] = 666;	Set the float before the memory block to 666. **VERY BAD.**
float *q = p + 10;	The pointer “q” now points to the 11th float in the memory.
q[-2] = 555.5;	The 9th float is set to 555.5. Valid, but uncommon.

The Golden Rule

```
float *p;
int i;

p[i] ≡ *(p+i)
```

Processes & Threads



Neil Coffey

top

```
top - 09:08:15 up 7 days, 22:56, 36 users,  load average: 4.03, 2.63, 1.78
Tasks: 852 total,   4 running, 848 sleeping,   0 stopped,   0 zombie
%Cpu(s):  5.8 us,  2.3 sy,  0.4 ni, 90.4 id,  0.9 wa,  0.0 hi,  0.2 si,  0.0 st
KiB Mem:  26368408+total, 19931880+used, 64365276 free,  5413456 buffers
KiB Swap: 13421772+total,     5144 used, 13421257+free. 14725691+cached Mem

  PID USER      PR  NI    VIRT    RES    SHR S   %CPU  %MEM     TIME+ COMMAND
30146 lyard     20   0  372268 248216   7768 R 147.06 0.094   4:23.17 python
 4608 wchen     20   0  707304 307296  21864 R 94.118 0.117   2:42.02 pdos.py
22076 dcampi    20   0  139080   7484   4432 R 82.353 0.003  31:12.44 sshd
 5549 dpotter   20   0   34592   4272   3068 R  5.882 0.002   0:00.01 top
25185 root      20   0       0      0      0 S  5.882 0.000  47:18.00 ptlrpcd_01_00
25186 root      20   0       0      0      0 S  5.882 0.000  43:56.62 ptlrpcd_01_01
25187 root      20   0       0      0      0 S  5.882 0.000  47:18.89 ptlrpcd_01_02
25188 root      20   0       0      0      0 S  5.882 0.000  43:50.86 ptlrpcd_01_03
25190 root      20   0       0      0      0 S  5.882 0.000  43:54.58 ptlrpcd_01_05
25193 root      20   0       0      0      0 S  5.882 0.000  47:13.45 ptlrpcd_01_08
25194 root      20   0       0      0      0 S  5.882 0.000  43:57.16 ptlrpcd_01_09
    1 root      20   0  120628   6636   4008 S  0.000 0.003  35:12.94 systemd
    2 root      20   0       0      0      0 S  0.000 0.000   0:31.76 kthreadd
    3 root      20   0       0      0      0 S  0.000 0.000   0:37.09 ksoftirqd/0
    5 root       0 -20       0      0      0 S  0.000 0.000   0:00.00 kworker/0:0H
    8 root      20   0       0      0      0 S  0.000 0.000  13:59.34 rcu_sched
    9 root      20   0       0      0      0 S  0.000 0.000   0:00.00 rcu_bh
   10 root      rt   0       0      0      0 S  0.000 0.000   0:05.07 migration/0
   11 root      rt   0       0      0      0 S  0.000 0.000   0:01.84 watchdog/0
```

top -u munge

```
top - 09:09:45 up 7 days, 22:58, 36 users,  load average: 4.37, 3.08, 2.02
Tasks: 847 total,   4 running, 843 sleeping,   0 stopped,   0 zombie
%Cpu(s):  5.8 us,  2.3 sy,  0.4 ni, 90.4 id,  0.9 wa,  0.0 hi,  0.2 si,  0.0 st
KiB Mem:  26368408+total, 19914214+used, 64541940 free,  5413456 buffers
KiB Swap: 13421772+total,     5144 used, 13421257+free. 14730092+cached Mem

  PID USER      PR  NI    VIRT    RES    SHR S   %CPU  %MEM     TIME+ COMMAND
18221 munge     20   0 2214036   5308   3404 S  0.000 0.002   1:16.34 munged
```

top -u munge -H

```
top - 09:10:29 up 7 days, 22:58, 36 users,  load average: 5.27, 3.45, 2.19
Threads: 1576 total,   7 running, 1569 sleeping,   0 stopped,   0 zombie
%Cpu(s): 12.8 us,  4.7 sy,  0.0 ni, 76.6 id,  5.6 wa,  0.0 hi,  0.3 si,  0.0 st
KiB Mem:  26368408+total, 19917673+used, 64507336 free,  5413456 buffers
KiB Swap: 13421772+total,     5144 used, 13421257+free. 14731705+cached Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
18221 munge     20   0 2214036   5308   3404 S 0.000 0.002   0:18.11 munged
18222 munge     20   0 2214036   5308   3404 S 0.000 0.002   0:02.86 munged
18223 munge     20   0 2214036   5308   3404 S 0.000 0.002   0:01.70 munged
18224 munge     20   0 2214036   5308   3404 S 0.000 0.002   0:01.66 munged
18225 munge     20   0 2214036   5308   3404 S 0.000 0.002   0:01.75 munged
18226 munge     20   0 2214036   5308   3404 S 0.000 0.002   0:01.72 munged
18227 munge     20   0 2214036   5308   3404 S 0.000 0.002   0:01.71 munged
18228 munge     20   0 2214036   5308   3404 S 0.000 0.002   0:01.74 munged
18229 munge     20   0 2214036   5308   3404 S 0.000 0.002   0:01.70 munged
18230 munge     20   0 2214036   5308   3404 S 0.000 0.002   0:01.72 munged
18231 munge     20   0 2214036   5308   3404 S 0.000 0.002   0:01.71 munged
18232 munge     20   0 2214036   5308   3404 S 0.000 0.002   0:01.74 munged
18233 munge     20   0 2214036   5308   3404 S 0.000 0.002   0:01.76 munged
18234 munge     20   0 2214036   5308   3404 S 0.000 0.002   0:01.71 munged
18235 munge     20   0 2214036   5308   3404 S 0.000 0.002   0:01.70 munged
18236 munge     20   0 2214036   5308   3404 S 0.000 0.002   0:01.73 munged
18237 munge     20   0 2214036   5308   3404 S 0.000 0.002   0:01.74 munged
18238 munge     20   0 2214036   5308   3404 S 0.000 0.002   0:01.72 munged
18239 munge     20   0 2214036   5308   3404 S 0.000 0.002   0:01.73 munged
```

Processes when you log in

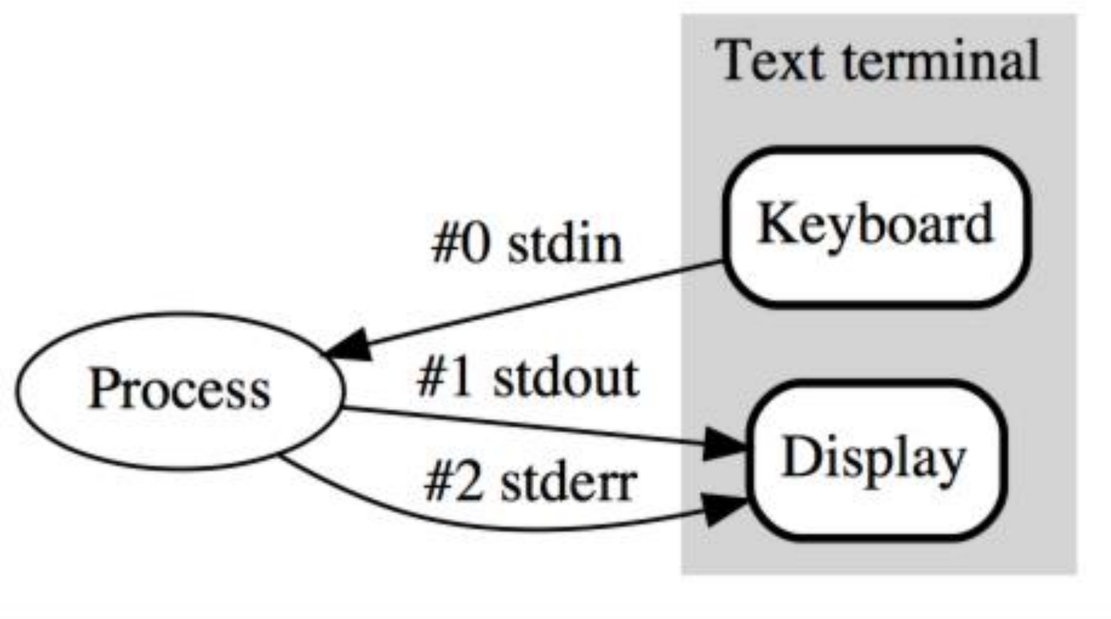
```
dpotter@daint102:~> ps -ef f
UID            PID    PPID  C STIME TTY        STAT    TIME CMD
root             1       0  0 Feb21 ?          Ss    579:14 /sbin/init
root         40965       1  0 Mar26 ?          Ss      0:22 /usr/sbin/sshd -D
root        138685   40965  0 Mar27 ?          Ss      0:00  \_ sshd: user [priv]
user        138688 138685  0 Mar27 ?          S       0:00  |   \_ sshd: user@notty
user        138689 138688  0 Mar27 ?          Ss      0:00  |       \_ /bin/someprog
root         27544   40965  0 18:31 ?          Ss      0:00  \_ sshd: dpotter [priv]
dpotter      27589   27544  0 18:31 ?          S       0:00      \_ sshd: dpotter@pts
dpotter      27590   27589  0 18:31 pts/21     Ss      0:00          \_ -bash
dpotter      28822   27590  0 18:31 pts/21     R+      0:00              \_ ps -ef f
```

/dev/tty	“Teletype” Terminal
/dev/pts	Pseudo-Terminal (pty) slave

A Brief History of Shells

- RUNCOM: 1963 – run list of commands (legacy /etc/rc*)
- Multics Shell: 1965 – a command launcher
- Thompson Shell: Ken Thompson 1971 – “sh”
  - More of a “command interpreter”
  - Introduced redirection and pipes (more on this later)
- Bourne Shell: Stephen Bourne (Bell Labs) 1979
  - Designed as a “scripting language”
- BASH: Brian Fox (GNU Project) 1989 – “bash”
  - Combines features in the Bourne Shell, “csh” and “ksh”
  - POSIX (Portable Operating System Interface) compliant
  - Now the de facto standard
  - Stands for “Bourne Again SHell” (ha ha ha)

Standard Files



Example: “cat”

```
dpotter@daint102:~> cat
I typed this line.
I typed this line.
This is the second line.
This is the second line.
<Control-D>
dpotter@daint102:~>
dpotter@daint102:~> <Control-D>
dpotter@daint102:~> logout
Connection to daint102 closed.
```

Must be at the start of the line of input

Redirection

```
ela2:> ls *.c
blaio.c  blas.c  cpi.c  foo.c  hello2.c  lapack.c  mdl.c  mpipi.c
res.c  rz.c  subarr.c  walk2.c  xthi.c


ela2:> ls *.c >output.dat
ela2:> cat output.dat
blaio.c
blas.c
cpi.c
foo.c
hello2.c
lapack.c
mdl.c
mpipi.c
res.c
rz.c
subarr.c
walk2.c
xthi.c
```

File Descriptors

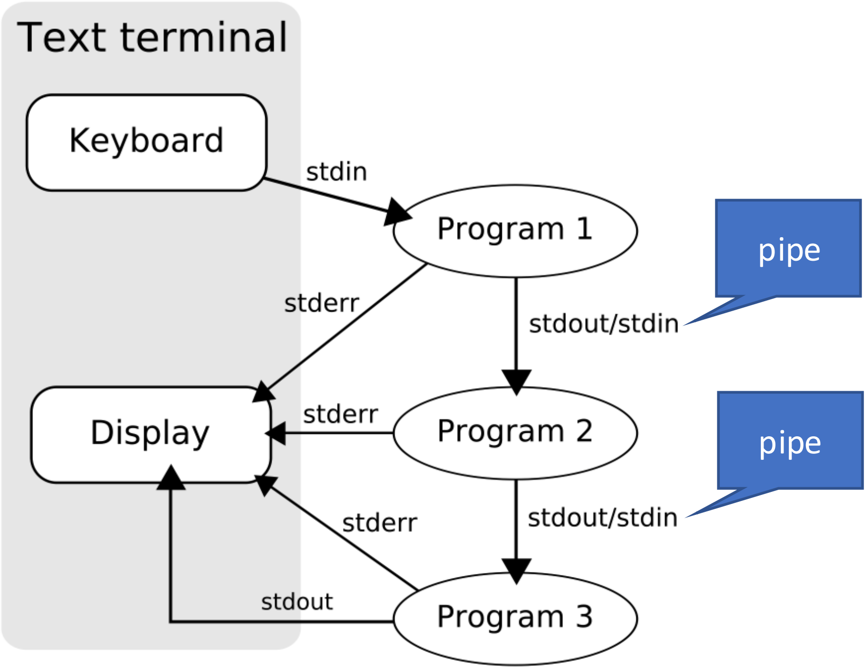
File Descriptor	Normal Target	>output.dat
0 (stdin)	/dev/tty	/dev/tty
1 (stdout)	/dev/tty	output.dat
2 (stderr)	/dev/tty	/dev/tty
3 (first program file)	not used until a file is opened by the program	

Redirect/To	stdin	stdout	stderr	“out.dat”
stdin	-	-	-	<out.dat
stdout	-	-	1>&2	>out.dat
stderr	-	2>&1	-	2>out.dat
stdout+stderr	-	-	-	&>out.dat

Appending to a file

```
ela3:> echo >data "This is one line"
ela3:> cat data
This is one line
ela3:> echo >data "This is line two"
ela3:> cat data
This is line two
ela3:> echo >>data "This is line three"
ela3:> cat data
This is line two
This is line three
```

Pipeline: program1 | program2 | program3



pipe

pipe

Example: “more”

```
dpotter@daint102:~/hpc1a> ps -ef | more
UID          PID    PPID  C STIME TTY          TIME CMD
root           1       0  0 Feb21 ?        09:33:20 /sbin/init
root           2       0  0 Feb21 ?        00:02:24 [kthreadd]
root           3       2  0 Feb21 ?        00:11:35 [ksoftirqd/0]
root           5       2  0 Feb21 ?        00:00:00 [kworker/0:0H]
root           8       2  0 Feb21 ?        01:28:45 [rcu_sched]
root           9       2  0 Feb21 ?        00:00:00 [rcu_bh]
root          10       2  0 Feb21 ?        00:00:56 [migration/0]
root          11       2  0 Feb21 ?        00:00:16 [watchdog/0]
root          12       2  0 Feb21 ?        00:00:14 [watchdog/1]
root          13       2  0 Feb21 ?        00:01:00 [migration/1]
root          14       2  0 Feb21 ?        00:04:36 [ksoftirqd/1]
root          18       2  0 Feb21 ?        00:00:15 [watchdog/2]
root          19       2  0 Feb21 ?        00:00:54 [migration/2]
root          20       2  0 Feb21 ?        01:46:42 [ksoftirqd/2]
root          22       2  0 Feb21 ?        00:00:00 [kworker/2:0H]
root          23       2  0 Feb21 ?        00:00:14 [watchdog/3]
root          24       2  0 Feb21 ?        00:00:47 [migration/3]
root          25       2  0 Feb21 ?        00:04:13 [ksoftirqd/3]
root          27       2  0 Feb21 ?        00:00:00 [kworker/3:0H]
root          28       2  0 Feb21 ?        00:00:15 [watchdog/4]
root          29       2  0 Feb21 ?        00:01:01 [migration/4]
root          30       2  0 Feb21 ?        00:11:45 [ksoftirqd/4]
root          32       2  0 Feb21 ?        00:00:00 [kworker/4:0H]
--More--
```

Example: “less”

```
dpotter@daint102:~/hpc1a> ps -ef | less
UID          PID    PPID  C STIME TTY          TIME CMD
root           1       0  0 Feb21 ?        09:33:20 /sbin/init
root           2       0  0 Feb21 ?        00:02:24 [kthreadd]
root           3       2  0 Feb21 ?        00:11:35 [ksoftirqd/0]
root           5       2  0 Feb21 ?        00:00:00 [kworker/0:0H]
root           8       2  0 Feb21 ?        01:28:45 [rcu_sched]
root           9       2  0 Feb21 ?        00:00:00 [rcu_bh]
root          10       2  0 Feb21 ?        00:00:56 [migration/0]
root          11       2  0 Feb21 ?        00:00:16 [watchdog/0]
root          12       2  0 Feb21 ?        00:00:14 [watchdog/1]
root          13       2  0 Feb21 ?        00:01:00 [migration/1]
root          14       2  0 Feb21 ?        00:04:36 [ksoftirqd/1]
root          18       2  0 Feb21 ?        00:00:15 [watchdog/2]
root          19       2  0 Feb21 ?        00:00:54 [migration/2]
root          20       2  0 Feb21 ?        01:46:42 [ksoftirqd/2]
root          22       2  0 Feb21 ?        00:00:00 [kworker/2:0H]
root          23       2  0 Feb21 ?        00:00:14 [watchdog/3]
root          24       2  0 Feb21 ?        00:00:47 [migration/3]
root          25       2  0 Feb21 ?        00:04:13 [ksoftirqd/3]
root          27       2  0 Feb21 ?        00:00:00 [kworker/3:0H]
root          28       2  0 Feb21 ?        00:00:15 [watchdog/4]
root          29       2  0 Feb21 ?        00:01:01 [migration/4]
root          30       2  0 Feb21 ?        00:11:45 [ksoftirqd/4]
root          32       2  0 Feb21 ?        00:00:00 [kworker/4:0H]
:
```

“less” is “more”

Who strive - you don't know how the others strive
To paint a little thing like that you smeared
Carelessly passing with your robes afloat,-
Yet do much less, so much less, Someone says,
(I know his name, no matter) - so much less!
Well, less is more, Lucrezia.

- *Andrea del Sarto*, 1855, Robert Browning

Example: My running processes

```
dpotter@daint102:~> ps -o pid,user,cmd -u dpotter f | less
 PID USER     CMD
 38794 dpotter  sshd: dpotter@pts/2
 38795 dpotter   \_ -bash
 41498 dpotter       \_ ps -o pid,user,cmd -u dpotter f
 41499 dpotter       \_ less
 … maybe other processes
(END)

daint104:> wc -l slurm-67584.out
144 slurm-67584.out

daint104:> less slurm-67584.out
daint104:> cat slurm-67584.out | less
           (obvious what happen here?)
```

Example: ls | wc

```
dpotter@daint102:~/hpc1a> ls
bad.c        cpi_openmp2.c   parse.pl
cpi          cpi_openmp.c    slurm-6188431.out
cpi.c        cpi_openmp.dat  slurm-6188433.out
cpi.eps      cpi_openmp.job  slurm-6192682.out
cpi_mpi      cpi.pdf         slurm-6312287.out
cpi_mpi.c    good.c          speedup.eps
cpi_mpi.job  good.s          speedup.gp
cpi_mpi.o    Makefile        speedup.pdf
cpi_openmp   old

dpotter@daint102:~/hpc1a> echo "just 3 words" | wc
      1       3      13

dpotter@daint102:~/hpc1a> echo "X" | wc
      1       1       2

dpotter@daint102:~/hpc1a> ls | wc
     26      26     283
```

Mystery Solved?

```
dpotter@daint102:~/hpc1a> ls | cat
bad.c
cpi
cpi.c
cpi.eps
cpi_mpi
cpi_mpi.c
cpi_mpi.job
cpi_mpi.o
cpi_openmp
cpi_openmp2.c
cpi_openmp.c
cpi_openmp.dat
cpi_openmp.job
cpi.pdf
good.c
good.s
Makefile
old
parse.pl
slurm-6188431.out
slurm-6188433.out
slurm-6192682.out
slurm-6312287.out
speedup.eps
speedup.gp
speedup.pdf
```

Redirection

```
dpotter@daint102:~/hpc1a> ls >foo.dat
dpotter@daint102:~/hpc1a> cat foo.dat
bad.c
cpi
cpi.c
cpi.eps
cpi_mpi
cpi_mpi.c
cpi_mpi.job
cpi_mpi.o
cpi_openmp
cpi_openmp2.c
cpi_openmp.c
cpi_openmp.dat
cpi_openmp.job
cpi.pdf
foo.dat
good.c
good.s
Makefile
old
parse.pl
slurm-6188431.out
slurm-6188433.out
slurm-6192682.out
slurm-6312287.out
speedup.eps
speedup.gp
speedup.pdf
dpotter@daint102:~/hpc1a> wc <foo.dat
 27  27 291
```

“Hello World” in C

```
#include <stdio.h>

int main(int argc,char *argv[]) {
    printf("Hello world\n");
    return 0;
    }
```

Arguments in C

```c
#include <stdio.h>

int main(int argc,char *argv[]) {
  int i;
  for(i=0; i<argc; ++i)
    printf("%d: %s\n",i,argv[i]);
  return 0;
  }
```

Python

```python
from sys import argv
for (i,v) in enumerate(argv):
    print("{}: {}".format(i,v))
```

Parameters

```
dpotter@daint103:~> ./hello2
0: ./hello2
dpotter@daint103:~> ./hello2 test a b c
0: ./hello2
1: test
2: a
3: b
4: c
dpotter@daint103:~> ./hello2 -h
0: ./hello2
1: -h
```

Wildcards (globbing)

```
daint103:~/hpc1a> ls *.c
bad.c  cpi.c  cpi_mpi.c  cpi_openmp2.c  cpi_openmp.c  good.c
daint103:~/hpc1a> ../hello2 *.c
0: ../hello2
1: bad.c
2: cpi.c
3: cpi_mpi.c
4: cpi_openmp2.c
5: cpi_openmp.c
6: good.c
daint103:~/hpc1a> ../hello2 "*.c"
0: ../hello2
1: *.c
daint103:~/hpc1a> ../hello2 '*.c'
0: ../hello2
1: *.c
daint103:~/hpc1a> ls "*.c"
ls: cannot access *.c: No such file or directory
```

Wildcards (globbing)

Wildcard	Description	Example
*	Matches zero or more	*.c (test.c example.c)
?	Matches exactly one	*.? (test.c foo.g)
[abc]	Only character in bracket	*.[ch]
[a-z]	Range of characters	[a-z]*, [a-zA-Z][0-9]*

man 7 glob

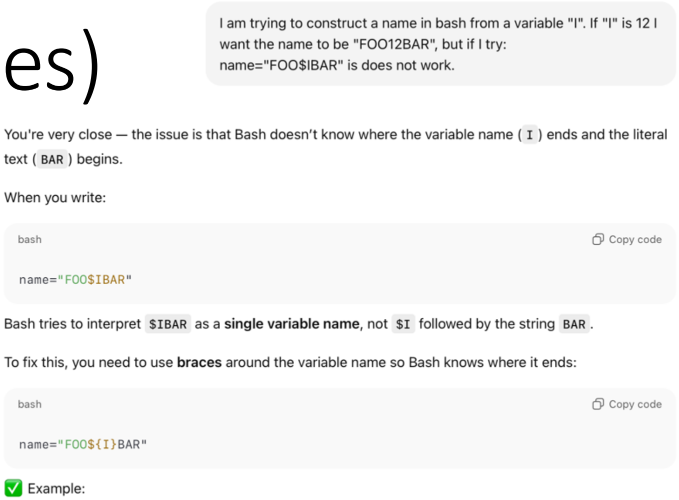
Expansion	Description	Example
{a,b,c}	Expands to the list of things	*.{c,h,cpp}
{s..e..i}	Counts from ‘s’ to ‘e’ in increments of ‘i’	{1..10..2}
~	Your home directory	~/.bashrc
~+	Your current working directory	

Variables (simple)

```
hpc:~$ TEST1=12
hpc:~$ set | grep TEST1
TEST1=12
hpc:~$ TEST2=string
hpc:~$ set | grep TEST2
TEST2=string
hpc:~$ TEST3=a long string
-bash: long: command not found
hpc:~$ TEST3="a long string"
hpc:~$ set | grep TEST3
TEST3='a long string'
```

Variables (using them and spaces)

```
hpc:~$ echo $TEST1
12
hpc:~$ echo $TEST3
a long string
hpc:~$ echo $TesT3
(nothing shows up here!)
hpc:~$ TEST4="a  long     string  with    spaces"
hpc:~$ echo ${TEST4}
a long string with spaces
hpc:~$ echo  this   has    space too
this has space too
```



I am trying to construct a name in bash from a variable "I". If "I" is 12 I want the name to be "FOO12BAR", but if I try: name="FOO$IBAR" is does not work.

You're very close — the issue is that Bash doesn't know where the variable name (`I`) ends and the literal text (`BAR`) begins.

When you write:

```bash
name="FOO$IBAR"
```

Bash tries to interpret `$IBAR` as a **single variable name**, not `$I` followed by the string `BAR`.

To fix this, you need to use **braces** around the variable name so Bash knows where it ends:

```bash
name="FOO${I}BAR"
```

✅ Example:

Variables (quoting)

```
hpc:~$ echo "$TEST1"
12
hpc:~$ echo '$TEST1'
$TEST1
hpc:~$ echo "$TEST4"
a  long     string  with    spaces
hpc:~$ echo "Answer is $TEST1"
Answer is 12
daint103:~/hpc1a> ../hello2  string   with   space
0: ../hello2
1: string
2: with
3: space
```

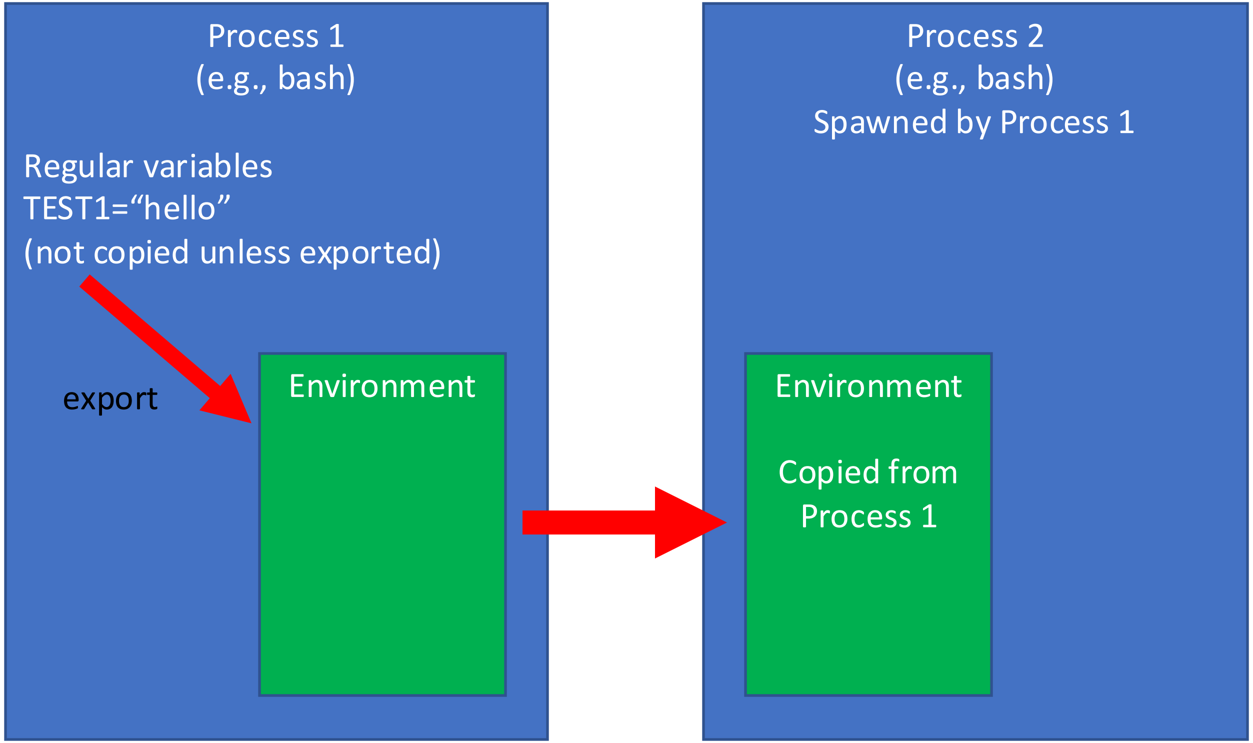
Environment Variables

```
dpotter@daint102:~> ps -o pid,command -u dpotter f
   PID COMMAND
 87343 sshd: dpotter@pts/51
 87345  \_ -bash
100054      \_ ps -o pid,command -u dpotter f

dpotter@daint102:~> bash

dpotter@daint102:~> ps -o pid,command -u dpotter f
   PID COMMAND
 87343 sshd: dpotter@pts/51
 87345  \_ -bash
106380      \_ bash
107695          \_ ps -o pid,command -u dpotter f
```

Environment Variables

```
dpotter@daint102:~> ETEST1="a variable"
dpotter@daint102:~> echo "$ETEST1"
a variable
dpotter@daint102:~> bash
dpotter@daint102:~> echo "$ETEST1"
(nothing shows up here!)
dpotter@daint102:~> exit
exit
dpotter@daint102:~> echo "$ETEST1"
a variable
dpotter@daint102:~> export ETEST1
dpotter@daint102:~> bash
dpotter@daint102:~> echo "$ETEST1"
a variable
dpotter@daint102:~> exit
```

Environment



Process 1
(e.g., bash)

Regular variables
TEST1="hello"
(not copied unless exported)

export

Environment

Process 2
(e.g., bash)
Spawned by Process 1

Environment

Copied from
Process 1

Environment variable “PATH”

```
dpotter@daint102:~> printenv PATH
/apps/uzh/daint/bin:/apps/uzh/bin:/users/dpotter/.local/bin:/users/dpotter/local/bin:/apps/daint/UES/xalt
/0.7.6/bin:/opt/slurm/17.11.12.cscs/bin:/opt/cray/pe/mpt/7.7.2/gni/bin:/opt/cray/pe/perftools/7.0.2/bin:/
opt/cray/pe/papi/5.6.0.2/bin:/opt/cray/rca/2.2.18-6.0.7.0_33.3__g2aa4f39.ari/bin:/opt/cray/alps/6.6.43-
6.0.7.0_26.4__ga796da3.ari/sbin:/opt/cray/job/2.2.3-
6.0.7.0_44.1__g6c4e934.ari/bin:/opt/cray/pe/craype/2.5.15/bin:/opt/cray/pe/cce/8.7.3/binutils/x86_64/x86_
64-pc-linux-gnu/bin:/opt/cray/pe/cce/8.7.3/binutils/cross/x86_64-aarch64/aarch64-linux-
gnu/../bin:/opt/cray/pe/cce/8.7.3/utils/x86_64/bin:/opt/cray/pe/modules/3.2.10.6/bin:/opt/slurm/default/b
in:/apps/daint/system/bin:/apps/common/system/bin:/usr/local/bin:/usr/bin:/bin:/usr/bin/X11:/usr/lib/mit/
bin:/usr/lib/mit/sbin:/opt/cray/pe/bin:/apps/dora/system/sbin
dpotter@daint102:~> export PATH=""
dpotter@daint102:~> ls
-bash: ls: No such file or directory
dpotter@daint102:~> top
-bash: top: No such file or directory
dpotter@daint102:~> /usr/bin/ls
… normal output from ls appears here
```

Scripting (bash is a program)

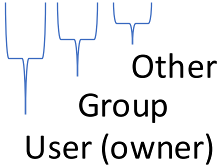
```
dpotter@daint102:~> cat script.sh
echo "This is a script"
echo "This does not do very much"
date
dpotter@daint102:~> bash script.sh
This is a script
This does not do very much
Mon Apr 17 17:16:03 CEST 2023
dpotter@daint102:~> bash <script.sh
This is a script
This does not do very much
Mon Apr 17 17:16:18 CEST 2023
dpotter@daint102:~> cat script.sh | bash
This is a script
This does not do very much
Mon Apr 17 17:16:33 CEST 2023
```

Scripting (Shebang)

```
dpotter@daint102:~> cat script.sh
#!/bin/bash
echo "This is a script"
echo "This does not do very much"
date
dpotter@daint102:~> chmod +x script.sh
dpotter@daint102:~> ./script.sh
This is a script
This does not do very much
Tue Apr 17 17:19:04 CEST 2018
dpotter@daint102:~> mv script.sh script
dpotter@daint102:~> ./script
This is a script
This does not do very much
Mon Apr 17 17:19:14 CEST 2023
```

File Permissions (a digression)

```
daint104:> ls -ld $HOME
drwx------  59 dpotter uzh4 12288 Apr 18 09:08 /users/dpotter
daint104:> ls -l test_file
-rw-r--r-- 1 dpotter uzh0 0 Apr 18 10:14 test_file
daint104:> chmod o-r test_file
daint104:> ls -l test_file
-rw-r----- 1 dpotter uzh0 0 Apr 18 10:14 test_file
daint104:> chmod 755 test_file
daint104:> ls -l test_file
-rwxr-xr-x 1 dpotter uzh0 0 Apr 18 10:14 test_file
```



Other

Group

User (owner)

```
daint104:> chmod g-rx,o-rx test_file
daint104:> ls -l test_file
-rwx------ 1 dpotter uzh0 0 Apr 18 10:14 test_file
```

Shebang Generally

```
dpotter@daint102:~> cat output
#!/usr/bin/cat
A Shebang program will run the program listed on the
first line, and it will "pipe" the file through it.
Usually this is used for programs that take commands,
but it is not restricted to this as you can see.
dpotter@daint102:~> ./output
#!/usr/bin/cat
A Shebang program will run the program listed on the
first line, and it will "pipe" the file through it.
Usually this is used for programs that take commands,
but it is not restricted to this as you can see.
```

Parameters in bash scripts

```
dpotter@daint102:~> cat script
#!/bin/bash
echo "Parameter 1: $1"
echo "Parameter 2: $2"
dpotter@daint102:~> ./script
Parameter 1:
Parameter 2:
dpotter@daint102:~> ./script one two three
Parameter 1: one
Parameter 2: two
dpotter@daint102:~> ./script "one two three" four
Parameter 1: one two three
Parameter 2: four
dpotter@daint102:~> ls *.c
blaio.c  blas.c  cpi.c  foo.c  hello2.c  lapack.c  mdl.c  mpipi.c
res.c  rz.c  subarr.c  walk2.c  xthi.c
dpotter@daint102:~> ./script *.c
Parameter 1: blaio.c
Parameter 2: blas.c
```

Parameters 0

```
dpotter@daint102:~> cat script0
#!/bin/bash
echo "Parameter 0: $0"
dpotter@daint102:~> ./script0
Parameter 0: ./script0
dpotter@daint102:~> ./script0 hello there
Parameter 0: ./script0
dpotter@daint102:~> cd code
dpotter@daint102:~/code> ../script0
Parameter 0: ../script0
dpotter@daint102:~> cd /tmp
dpotter@daint102:/tmp> /users/dpotter/script0
Parameter 0: /users/dpotter/script0

dpotter@daint102:~/code> script0
Parameter 0: /users/dpotter/.local/bin/script0
```

“simulate”

```
daint102:>./simulate
How many stacks do you need?
9
What is the mean density?
1.3
Seed?
314159365
Grid size?
256
Generating stacks...
done.
Wallclock 1h45m12s
```

“simulate” Problems

- We would like to run this 100 times
  - Different set of parameters for each run
  - Independent jobs
- How can we feed in the variables?
  - Edit a file and pipe it in (or redirection)

    ```
    ./simulate <parameters
    ```
  - Create the file on-the-fly with “echo”
  - There is a better way though

“HERE” Documents

```
daint102:> cat script
#!/bin/bash
cat <<EOF
This is a line of text.
Here is another.
EOF
wc <<EOF
Input can go to any program.
We have three lines here.
And 14 words.
EOF
daint102:> ./script
This is a line of text.
Here is another.
 3 14 69
```

“HERE” Documents

```
daint102:> cat script
#!/bin/bash
A="the variable A"
cat <<WHATEVER
Even more impressive is
that we can have variables.

Parameter 1: $1
Variable A: $A
WHATEVER
daint102:> ./script "parameter 1"
Even more impressive is
that we can have variables.

Parameter 1: parameter 1
Variable A: the variable A
```

“simulate” 1.0

```
#!/bin/bash
STACKS=9
DENSITY=1.3
SEED=314159265
GRID=256
./simulate <<EOD
$STACKS
$DENSITY
$SEED
$GRID
EOD
```

“simulate” 2.0

```
#!/bin/bash
STACKS=$1
DENSITY=$2
SEED=$3
GRID=$4
./simulate <<EOD
$STACKS
$DENSITY
$SEED
$GRID
EOD
```

The “if/then/else” statement

```
if <some condition> then
    <take some action>
end if

if (i < 100) {
    printf("%d\n”, i);
}

if i < 100:
    print(i)

BASH???

if $A < 100 then . ✗
```

“If/then” in bash

```
if <command line>
then
  <command line(s)>
fi

if date
then
  echo "This was true???"
fi
dpotter@daint102:~> if date
> then
>   echo "This was true???"
> fi
Tue Apr 17 17:34:25 CEST 2018
This was true???
```

“If/then” in bash

```
command1 ; command2 ; command3 ; …

if <command line> ; then <command> ; fi

daint102:~> true
daint102:~> false
daint102:~> if true ; then echo yes ; fi
yes
daint102:~> if false ; then echo yes ; fi
daint102:~>
```

“If/then/else if/else”

```
if <command line> ; then
  <command lines>
elif <command line> ; then
  <command lines>
elif <command line> ; then
  <command lines>
elif <command line> ; then
  <command lines>
else
  <command lines>
fi
```

“Hello World” in C

```c
#include <stdio.h>

int main(int argc,char *argv[]) {
    printf("Hello world\n");
    return 0;
    }
```

The Return Code (error code)

```
daint102:~/hpc1a> ls >/dev/null
daint102:~/hpc1a> echo $?
0
daint102:~/hpc1a> ls >/dev/null no_such_file
ls: cannot access 'no_such_file': No such file or directory
daint102:~/hpc1a> echo $?
2
daint102:~/hpc1a> echo $?
0
daint102:~/hpc1a> true ; echo $?
0
daint102:~/hpc1a> false ; echo $?
1
```

“Boolean” Logic

```
daint102:> if true ; then echo yes ; fi
yes
daint102:> if true && true ; then echo yes ; fi
yes
daint102:> if true && false ; then echo yes ; fi
daint102:> if true || false ; then echo yes ; fi
yes
daint102:> if false || false ; then echo yes ; fi
daint102:> if true && false || true ; then echo yes ; fi
yes
daint102:> if true || false && true ; then echo yes ; fi
yes
daint102:> if true && echo second ; then echo yes ; fi
second
yes
daint102:> if true || echo second ; then echo yes ; fi
yes
```

Short circuit

test, [, [[

```
daint102:> A=12
daint102:> test $A -gt 10 ; echo $?
0
daint102:> test $A -gt 20 ; echo $?
1
daint102:> if test $A -gt 10 ; then echo large ; fi
large
daint102:> if [ $A -gt 10 ] ; then echo large ; fi
large
daint102:> if [[ $A -gt 10 ]] ; then echo large ; fi
large
daint102:> if [[ $A > 10 ]] ; then echo large ; fi
large
daint102:> if [[ $A > 20 ]] ; then echo large ; fi
daint102:> if [[ $A > 101 ]] ; then echo large ; fi
large
```

Lexicographical (Alphabetical)

“while”

```
while <command line>
do
  <command line>
done
daint102:> cat while.sh
#!/bin/bash
i=0
while test $i -lt 5 ; do
  echo $i
  let ++i
done
daint102:> ./while.sh
0
1
2
3
4
```

“for”

```
for NAME in word
do
  <command line>
done
daint104:> for V in a b c ; do echo $V ; done
a
b
c
daint104:> ls slurm-6*
slurm-67584.out  slurm-67594.out  slurm-67640.out  slurm-
6878386.out
daint104:> for NAME in slurm-6* ; do echo $NAME ; done
slurm-67584.out
slurm-67594.out
slurm-67640.out
slurm-6878386.out
daint104:> for (( i=0; i<10 ; ++i )) ; do echo $i ; done
0
1
…
9
```

“shift”

```
daint102:> cat shift
#!/bin/bash
while test -n "$1" ; do
  echo "Processing $1"
  shift
done
daint102:> ./shift
daint102:> ls *.c
blaio.c  blas.c  cpi.c  foo.c  hello2.c  lapack.c  mdl.c
mpipi.c  res.c  rz.c  subarr.c  walk2.c  xthi.c
daint102:> ./shift *.c
Processing blaio.c
Processing blas.c
Processing cpi.c
Processing foo.c
Processing hello2.c
Processing lapack.c
Processing mdl.c
Processing mpipi.c
Processing res.c
Processing rz.c
Processing subarr.c
Processing walk2.c
Processing xthi.c
```

Integer Arithmetic $(( )) – also (())

```
ela3:~> A=90
ela3:~> B=$((A/2+7))
ela3:~> echo $B
52
ela3:~> C=$((B*2+A))
ela3:~> echo $C
194
ela3:~> D=$((C/5))
ela3:~> echo $D
38
```

Real Arithmetic ???

```
ela3:~> A=$((30.2))
-bash: 30.2: syntax error: invalid arithmetic operator (error
token is ".2")
ela3:~> A=30.2
ela3:~> B=12.4
ela3:~> C=$((A+B))
-bash: 30.2: syntax error: invalid arithmetic operator (error
token is ".2")
```

Well, drat!

Real Arithmetic – not possible?

```
daint103:> bc
bc 1.06.95
Copyright 1991-1994, 1997, 1998, 2000, 2004, 2006 Free Software
Foundation, Inc.
This is free software with ABSOLUTELY NO WARRANTY.
For details type `warranty'.
1/3
0
scale=10
1/3
.3333333333
scale=9;1/3
.333333333
daint103:> echo "scale=10;1.0/3.0" | bc
.3333333333
```

Output Capture $( ), ` `

```
daint103:> ANSWER=$(echo "scale=10;1.0/3.0" | bc)
daint103:> echo $ANSWER
.3333333333
daint103:> RESULT=`echo "scale=10;1.0/3.0" | bc`
daint103:> echo $RESULT
.3333333333
daint103:> ls *.c
blaio.c  blas.c  cpi.c  foo.c  hello2.c  lapack.c
mdl.c  mpipi.c  res.c  rz.c  subarr.c  walk2.c  xthi.c
daint103:> FILES=$(ls *.c)
daint103:> echo $FILES
blaio.c blas.c cpi.c foo.c hello2.c lapack.c mdl.c mpipi.c
res.c rz.c subarr.c walk2.c xthi.c
```

Functions

```
daint103:> cat distance
#!/bin/bash
function distance () {
  local x=$1
  local y=$2
  echo "scale=10;sqrt($x*$x + $y*$y)" | bc
  return 0
}
distance 3 4
distance 10 9
daint103:> ./distance
5.0000000000
13.4536240470
```

Function Capture

```
daint103:> cat distance
#!/bin/bash
function distance () {
  local x=$1
  local y=$2
  echo "scale=10;sqrt($x*$x + $y*$y)" | bc
}
D1=$(distance 3 4)
D2=$(distance 10 9)
echo "Distance 1 is $D1"
daint103:> ./distance
Distance 1 is 5.0000000000
```

“simulate” revisited

```
daint103:> cat script.job
#!/bin/bash
function distance () {
  local x=$1
  local y=$2
  echo "scale=10;sqrt($x*$x + $y*$y)" | bc
}
cat <<EOF
9
$(distance 3 4)
314159265
$(distance 10 9)
EOF
ela3:> ./script.job
9
5.0000000000
314159265
13.4536240470
```

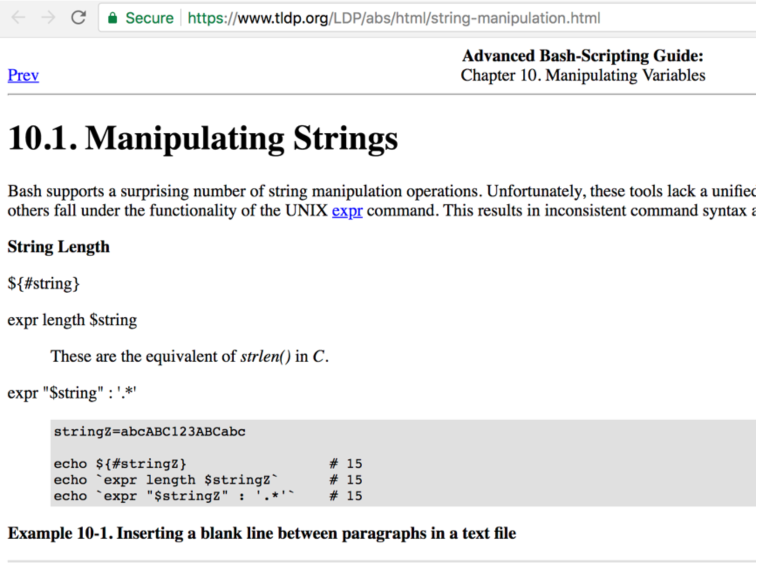
“cat”, but could be “simulate”

Functions return values

```
daint103:> cat prime
#!/bin/bash
function isprime () {
  # should check if $1 is prime, but:
  return 0
}
if isprime $1 ; then
  echo "$1 is prime”
else
  echo "$1 is not prime”
fi
daint103:> ./prime 12
12 is prime
```

Advanced Features

- BASH has arrays as well
- String Manipulation



Prev

Advanced Bash-Scripting Guide:
Chapter 10. Manipulating Variables

10.1. Manipulating Strings

Bash supports a surprising number of string manipulation operations. Unfortunately, these tools lack a unifiec
others fall under the functionality of the UNIX expr command. This results in inconsistent command syntax a

String Length

${#string}

expr length $string

These are the equivalent of *strlen()* in *C*.

expr "$string" : '.*'

```
stringZ=abcABC123ABCabc

echo ${#stringZ}                 # 15
echo `expr length $stringZ`      # 15
echo `expr "$stringZ" : '.*'`    # 15
```

Example 10-1. Inserting a blank line between paragraphs in a text file

Grep: Searching in files

```
hpc:~$ cat interim.txt
Interim
Lola Ridge, 1873

The earth is motionless
And poised in space ...
A great bird resting in its flight
Between the alleys of the stars.
It is the wind’s hour off ...
The wind has nestled down among the corn ...
The two speak privately together,
Awaiting the whirr of wings.
hpc:~$ grep alley interim.txt
Between the alleys of the stars.
```

Grep: Searching in files

```
hpc:~$ grep --color alley interim.txt
Between the alleys of the stars.
hpc:~$ grep --color the interim.txt
Between the alleys of the stars.
It is the wind’s hour off ...
The wind has nestled down among the corn ...
The two speak privately together,
Awaiting the whirr of wings.
hpc:~$ grep -c the interim.txt
5
hpc:~$ grep -q the interim.txt
hpc:~$ echo $?
0
```

Grep: Searching in files

```
hpc:~$ if grep -q alley interim.txt ; then
> echo found
> fi
found
hpc:~$ if grep -q apple interim.txt ; then
> echo found
> fi
hpc:~$ grep -q apple interim.txt
hpc:~$ echo $?
1
```

Grep: Some useful flags

```
hpc:~$ grep --color and interim.txt
hpc:~$ grep --color -i and interim.txt
And poised in space ...
hpc:~$ grep --color -i -A 1 and interim.txt
And poised in space ...
A great bird resting in its flight
hpc:~$ grep --color -i -B 1 and interim.txt
The earth is motionless
And poised in space ...
hpc:~$ grep --color -i -C 1 and interim.txt
The earth is motionless
And poised in space ...
A great bird resting in its flight
```

Global Regular Expression Print

- Appear in Version 4 Unix in 1973
- Created by Ken Thompson (original Unix)
- Based on a regular expression parser in “ed”
- “g/re/p” command
  - /re/ – Search for a regular expression “re”
  - g – do it globally
  - p – print the result
- Hence: g/re/p → grep!

Global Regular Expression Print

```
hpc:~$ grep --color a..ey interim.txt
Between the alleys of the stars.
hpc:~$ grep --color all. interim.txt
Between the alleys of the stars.
hpc:~$ grep --color all.* interim.txt
Between the alleys of the stars.
hpc:~$ grep --color al*ey interim.txt
Between the alleys of the stars.
hpc:~$ grep --color we*n interim.txt
Between the alleys of the stars.
The wind has nestled down among the corn ...
```

Extended **G R E P**

```
hpc:~$ egrep --color we*n interim.txt
Between the alleys of the stars.
The wind has nestled down among the corn ...
hpc:~$ egrep --color we+n interim.txt
Between the alleys of the stars.
hpc:~$ egrep --color wee*n interim.txt
Between the alleys of the stars.
hpc:~$ egrep --color we?x?e?v?n interim.txt
Between the alleys of the stars.
The wind has nestled down among the corn ...
```

Extended G R E P

```
hpc:~$ export GREP_OPTIONS=--color
hpc:~$ egrep -i the interim.txt
The earth is motionless
Between the alleys of the stars.
It is the wind’s hour off ...
The wind has nestled down among the corn ...
The two speak privately together,
Awaiting the whirr of wings.
hpc:~$ egrep -i 'the wind' interim.txt
It is the wind’s hour off ...
The wind has nestled down among the corn ...
```

Greedy Matching (the default)

```
hpc:~$ egrep 'alleys' interim.txt
Between the alleys of the stars.

hpc:~$ egrep 'all.*s' interim.txt
Between the alleys of the stars.
```

Greedy Match

```
hpc:~$ egrep 'all.*?s' interim.txt
Between the alleys of the stars.
```

Minimum Match

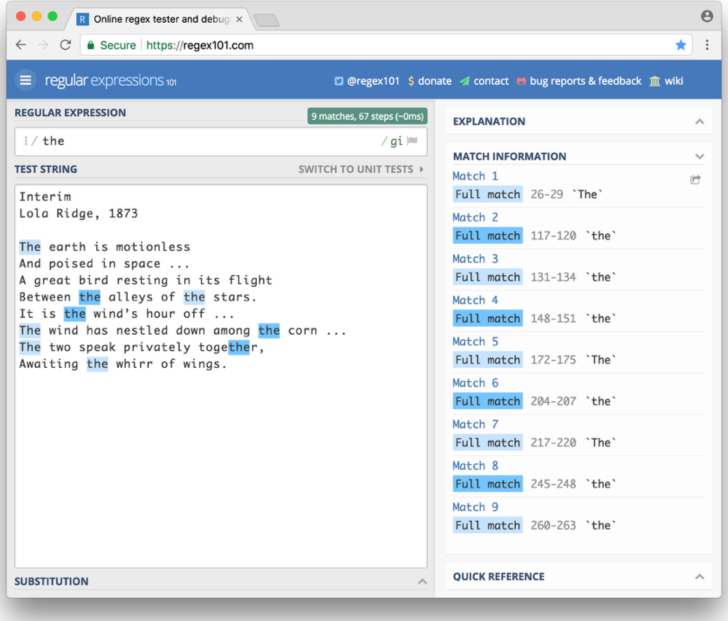
Alternate values (or)

```
hpc:~$ export GREP_OPTIONS='-i --color'
hpc:~$ egrep 'the wind' interim.txt
It is the wind’s hour off ...
The wind has nestled down among the corn ...
hpc:~$ egrep 'the earth' interim.txt
The earth is motionless
hpc:~$ egrep 'the wind|the earth' interim.txt
The earth is motionless
It is the wind’s hour off ...
The wind has nestled down among the corn ...
```

Parenthesis

```
hpc:~$ export GREP_OPTIONS='-i --color'
hpc:~$ egrep 'the wind|earth' interim.txt
The earth is motionless
It is the wind’s hour off ...
The wind has nestled down among the corn ...
hpc:~$ egrep 'the (wind|earth)' interim.txt
The earth is motionless
It is the wind’s hour off ...
The wind has nestled down among the corn ...
```

regex101.com


Online regex tester and debug ×
Secure | https://regex101.com
regular expressions 101
@regex101 $ donate contact bug reports & feedback wiki
SWITCH TO UNIT TESTS ›
EXPLANATION
SUBSTITUTION
QUICK REFERENCE

REGULAR EXPRESSION — 9 matches, 67 steps (~0ms)

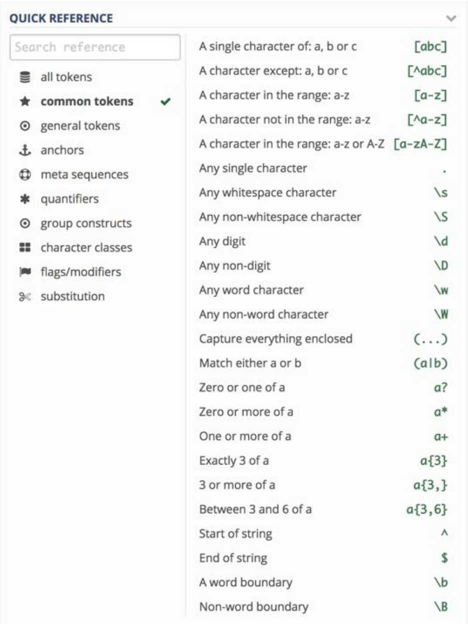
```
/ the /gi
```

TEST STRING

```
Interim
Lola Ridge, 1873

The earth is motionless
And poised in space ...
A great bird resting in its flight
Between the alleys of the stars.
It is the wind’s hour off ...
The wind has nestled down among the corn ...
The two speak privately together,
Awaiting the whirr of wings.
```

MATCH INFORMATION

```
Match 1
Full match  26-29   `The`
Match 2
Full match  117-120 `the`
Match 3
Full match  131-134 `the`
Match 4
Full match  148-151 `the`
Match 5
Full match  172-175 `The`
Match 6
Full match  204-207 `the`
Match 7
Full match  217-220 `The`
Match 8
Full match  245-248 `the`
Match 9
Full match  260-263 `the`
```



A single character of: a, b or c	[abc]
A character except: a, b or c	[^abc]
A character in the range: a-z	[a-z]
A character not in the range: a-z	[^a-z]
A character in the range: a-z or A-Z	[a-zA-Z]
Any single character	.
Any whitespace character	\s
Any non-whitespace character	\S
Any digit	\d
Any non-digit	\D
Any word character	\w
Any non-word character	\W
Capture everything enclosed	(...)
Match either a or b	(a|b)
Zero or one of a	a?
Zero or more of a	a*
One or more of a	a+
Exactly 3 of a	a{3}
3 or more of a	a{3,}
Between 3 and 6 of a	a{3,6}
Start of string	^
End of string	$
A word boundary	\b
Non-word boundary	\B

BASH Regular Expressions

```
hpc:~$ cat retest.sh
#!/bin/bash
while read A ; do
  echo "$A"
done
hpc:~$ ./retest.sh <interim.txt
Interim
Lola Ridge, 1873

The earth is motionless
And poised in space ...
A great bird resting in its flight
Between the alleys of the stars.
It is the wind’s hour off ...
The wind has nestled down among the corn ...
The two speak privately together,
Awaiting the whirr of wings.
```

BASH Regular Expressions

```
hpc:~$ cat retest.sh
#!/bin/bash
while read A ; do
  if [[ "$A" =~ we*n ]] ; then
    echo "$A"
  fi
done
hpc:~$ ./retest.sh <interim.txt
Between the alleys of the stars.
The wind has nestled down among the corn ...
hpc:~$ [[ "Hello world" =~ music ]] ; echo $?
1
hpc:~$ [[ "Hello world" =~ ll ]] ; echo $?
0
```

BASH Regular Expressions

```
hpc:~$ cat report.txt
Starting calculations
Phase 1 complete, 3.11 seconds
Phase 2 complete, 10.92 seconds
Phase 3 complete, 1.15 seconds
Calculations complete
hpc:~$ egrep 'Phase [0-9] complete, [0-9]+\.[0-
9]+ seconds' report.txt
Phase 1 complete, 3.11 seconds
Phase 2 complete, 10.92 seconds
Phase 3 complete, 1.15 seconds
```

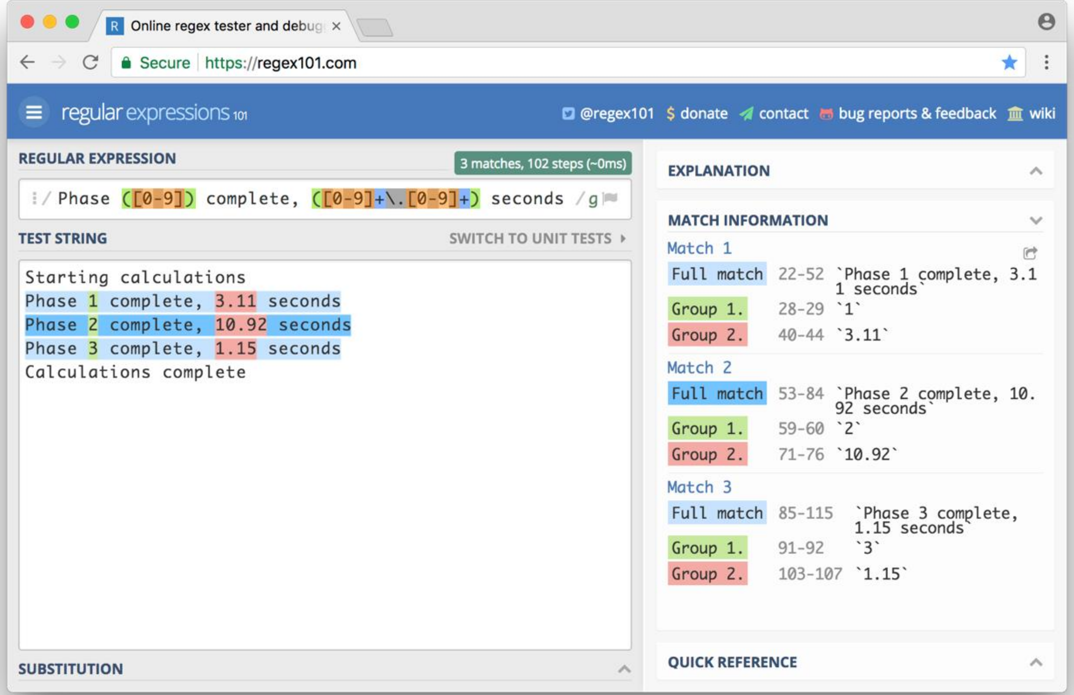
BASH Regular Expressions

```
hpc:~$ cat retest.sh
#!/bin/bash
RE="Phase [0-9] complete, [0-9]+\.[0-9]+
seconds"
while read A ; do
  if [[ "$A" =~ $RE ]] ; then
    echo "$A"
  fi
done
hpc:~$ ./retest.sh <report.txt
Phase 1 complete, 3.11 seconds
Phase 2 complete, 10.92 seconds
Phase 3 complete, 1.15 seconds
```

Regular Expression Capture

```
hpc:~$ cat retest.sh
#!/bin/bash
RE="Phase ([0-9]) complete, ([0-9]+\.[0-9]+)
seconds"
while read A ; do
  if [[ "$A" =~ $RE ]] ; then
    echo "Match: ${BASH_REMATCH[0]}"
    echo "Time:  ${BASH_REMATCH[2]}"
  fi
done
hpc:~$ ./retest.sh <report.txt
Match: Phase 1 complete, 3.11 seconds
Time:  3.11
Match: Phase 2 complete, 10.92 seconds
Time:  10.92
Match: Phase 3 complete, 1.15 seconds
Time:  1.15
```

Group matches



Online regex tester and debug
Secure https://regex101.com
regular expressions 101
@regex101 $ donate contact bug reports & feedback wiki
SWITCH TO UNIT TESTS
SUBSTITUTION
QUICK REFERENCE

REGULAR EXPRESSION 3 matches, 102 steps (~0ms)

```
/ Phase ([0-9]) complete, ([0-9]+\.[0-9]+) seconds /g
```

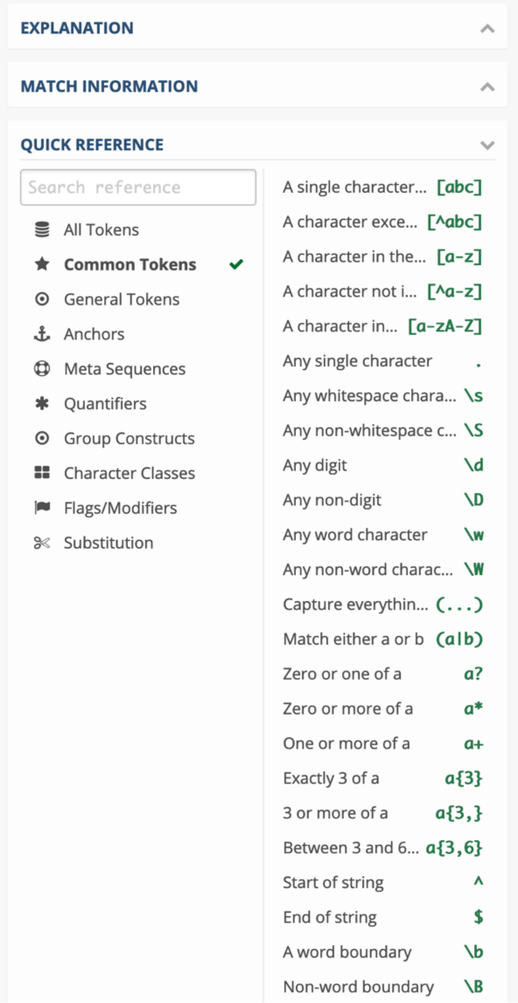
TEST STRING

```
Starting calculations
Phase 1 complete, 3.11 seconds
Phase 2 complete, 10.92 seconds
Phase 3 complete, 1.15 seconds
Calculations complete
```

EXPLANATION

MATCH INFORMATION

```
Match 1
Full match  22-52    `Phase 1 complete, 3.11 seconds`
Group 1.    28-29    `1`
Group 2.    40-44    `3.11`
Match 2
Full match  53-84    `Phase 2 complete, 10.92 seconds`
Group 1.    59-60    `2`
Group 2.    71-76    `10.92`
Match 3
Full match  85-115   `Phase 3 complete, 1.15 seconds`
Group 1.    91-92    `3`
Group 2.    103-107  `1.15`
```



A single character...	[abc]
A character exce...	[^abc]
A character in the...	[a-z]
A character not i...	[^a-z]
A character in...	[a-zA-Z]
Any single character	.
Any whitespace chara...	\s
Any non-whitespace c...	\S
Any digit	\d
Any non-digit	\D
Any word character	\w
Any non-word charac...	\W
Capture everythin...	(...)
Match either a or b	(a|b)
Zero or one of a	a?
Zero or more of a	a*
One or more of a	a+
Exactly 3 of a	a{3}
3 or more of a	a{3,}
Between 3 and 6...	a{3,6}
Start of string	^
End of string	$
A word boundary	\b
Non-word boundary	\B

Regular Expression Capture

```
hpc:~$ cat retest.sh
#!/bin/bash
RE="Phase ([0-9]) complete, ([0-9]+\.[0-9]+)
seconds"
TIME=0
while read A ; do
  if [[ "$A" =~ $RE ]] ; then
    TIME=$(echo "scale=5;$TIME +\
            ${BASH_REMATCH[2]}" | bc)
  fi
done
echo "Total time: $TIME seconds"
hpc:~$ ./retest.sh <report.txt
Total time: 15.18 seconds
```

University of Zurich^UZH^

Python

```
hpc:~$ cat retest.py
#!/usr/bin/python
import re
import fileinput
pattern = re.compile(
  'Phase ([0-9]) complete, ([0-9]+\.[0-9]+) seconds')
time = 0
for line in fileinput.input():
  match = pattern.match(line)# None or “Match Object”
  if match:
    time += float(match.group(2))
print("Total time: %f seconds" % time)
hpc:~$ ./retest.py <report.txt
Total time: 15.180000 seconds
```