CS 241: Systems Programming Lecture 28. Signals Fall 2019 Prof. Stephen Checkoway

Which of the following is the standard procedure to run a new program cmd with argument arg (error checking has been omitted below)?

- A. pid t pid = fork(); if (pid == 0) execl(path, path, arg, (char *)0);
- B. pid t pid = fork(); if (pid != 0) execl(path, path, arg, (char *)0);
- C. int ret = execl(path, path, arg, (char *)0); **if** (ret == 0) fork();
- D. int ret = execl(path, path, arg, (char *)0); **if** (ret != 0) fork();

Redirection

From last class's exercises, the dup2 system call creates a new file descriptor that refers to the same file as the old descriptor

int dup2(int oldfd, int newfd);

- We can use this to perform redirection of stdin/stdout/stderr 1. Open the file we want to redirect input from/output to 2. dup2 the returned file descriptor to STDIN FILENO/STDOUT FILENO/
 - **STDERR FILENO**
 - 3. Close the original file descriptor

Strace example

When running

\$ /bin/echo hello > output.txt

On Linux, fork() uses the clone system call

- clone(...) = 80398038
- 8039
- dup2(3, 1) 8039
- close(3)8039
- execve("/bin/echo", ["/bin/echo", "hello"], ...) = 0 8039

- we can see the sequence of system calls Bash makes using strace -f

```
openat(AT_FDCWD, "output.txt", O_WRONLY O_CREAT O_TRUNC, 0666) = 3
                                   = 1
                                   = 0
```



What about pipes?

\$ /bin/echo hi | /usr/bin/head

We can strace this!



<pre>execve("/bin/bash", ["bash", "-c"</pre>
<pre>pipe([3, 4])</pre>
clone()
close(4)
clone()
close(3)
close(3)
<pre>wait4(-1, <unfinished></unfinished></pre>
dup2(4, 1)
dup2(3, 0)
close(3)
close(4)
<pre>execve("/usr/bin/head", ["/usr/bit</pre>
<pre>execve("/bin/echo", ["/bin/echo",</pre>
< wait4 resumed> [{WIFEXITED(s
<pre>wait4(-1, <unfinished></unfinished></pre>
< wait4 resumed> [{WIFEXITED(s

12449 – ba	S	h
------------	---	---

- echo
- 12451 head





Pipes

Oldest form of UNIX System interprocess communication (IPC)

Have some limitations:

- - Data only flows one direction
 - Some systems have full-duplex, but this isn't standard
- Only can be used between processes with a common ancestor

Historically have been half-duplex (data only flows one direction)

pipe(2)

#include <unistd.h>

- int pipe(int fd[2])
 - Returns 0 on success, -1 on error

Returns values in array fd

- fd[0] is opened for reading
- fd[1] is opened for writing

File descriptors are connected to each other!

After call to pipe()



or



From Advanced Programming in the UNIX[®] Environment, Third Edition, by W. Richard Stevens and Stephen A. Rago (ISBN-13: 978-0-321-63773-4). Copyright © 2013 by Pearson Education, Inc. All rights reserved.



A pipe in a single process is usually unnecessary We can already talk to ourselves!

Normally, you create a pipe and then fork()

This creates a channel from parent to child (or vice versa)

After call to fork()



Figure 15.3 Half-duplex pipe after a fork

From Advanced Programming in the UNIX[®] Environment, Third Edition, by W. Richard Stevens and Stephen A. Rago (ISBN-13: 978-0-321-63773-4). Copyright © 2013 by Pearson Education, Inc. All rights reserved.

kernel

Close unneeded descriptors



Figure 15.4 Pipe from parent to child

From Advanced Programming in the UNIX[®] Environment, Third Edition, by W. Richard Stevens and Stephen A. Rago (ISBN-13: 978-0-321-63773-4). Copyright © 2013 by Pearson Education, Inc. All rights reserved.

12449	<pre>execve("/bin/bash", ["bash", "-c"</pre>
12449	pipe([3, 4])
12449	clone()
12449	close(4)
12449	clone()
12449	close(3)
12450	close(3)
12449	<pre>wait4(-1, <unfinished></unfinished></pre>
12450	dup2(4, 1)
12451	dup2(3, 0)
12451	close(3)
12450	close(4)
12451	<pre>execve("/usr/bin/head", ["/usr/bit</pre>
12450	<pre>execve("/bin/echo", ["/bin/echo",</pre>
12449	< wait4 resumed> [{WIFEXITED(s
12449	<pre>wait4(-1, <unfinished></unfinished></pre>
12449	< wait4 resumed> [{WIFEXITED(s

12449 – basł

- **12450** echo
- 12451 head

			_						
7	"/	bin/echo	hi	/usr	/bin/l	nead	1"],)) =
=	= 0								
-	= 1	2450	fd	stra	ce'				
=	= 0		0	torm	inal				
=	= 1	2451	U	lenn	IIIai				
=	= 0		1	term	inal				
-	= 0			_					
			2	term	inal				
=	= 1		3						
=	= 0		0						
=	= 0		4						
=	= 0								
n,	/he	ad"],	.) = 0						
	"hi)	= 0						
)	& &	WEXITST	ATUS (S) ==	0}],	0,	NULL)) =	124
)	& &	WEXITST	ATUS (S) ==	0}],	0,	NULL)) =	124









12449	<pre>execve("/bin/bash", ["bash",</pre>	"-c", "/bin/e	cho hi	/usr/bin/l	nead"], .) =
12449	pipe([3, 4])	= 0				
12449	clone()	= 12450	fd	bash		
12449	close(4)	= 0				
12449	clone()	= 12451	0	terminal		
12449	close(3)	= 0	1	terminal		
12450	close(3)	= 0		Commu		
12449	<pre>wait4(-1, <unfinished></unfinished></pre>		2	terminal		
12450	dup2(4, 1)	= 1	•			
12451	dup2(3, 0)	= 0	3			
12451	close(3)	= 0	4			
12450	close(4)	= 0				
12451	<pre>execve("/usr/bin/head", ["/u</pre>	<pre>sr/bin/head"],</pre>) = 0			
12450	<pre>execve("/bin/echo", ["/bin/e</pre>	cho", "hi"], .) = 0			
12449	< wait4 resumed> [{WIFEXI	TED(S) && WEXI	TSTATUS (S	$) == 0 \}],$	0, NULL)	= 124
12449	<pre>wait4(-1, <unfinished></unfinished></pre>					
12449	< wait4 resumed> [{WIFEXI	TED(S) && WEXI	TSTATUS (S) == 0}],	0, NULL)	= 124

12449 - ba	ash
------------	-----

- **12450 echo**
- 12451 head









<pre>cve("/bin/bash", ["bash", "-c" e([3, 4])</pre>
e([3, 4])
ne()
se(4)
ne()
se(3)
se(3)
t4(-1, <unfinished></unfinished>
2(4, 1)
2(3, 0)
se(3)
se(4)
<pre>cve("/usr/bin/head", ["/usr/bi</pre>
<pre>cve("/bin/echo", ["/bin/echo",</pre>
<pre>. wait4 resumed> [{WIFEXITED(s</pre>
t4(-1, <unfinished></unfinished>
. wait4 resumed> [{WIFEXITED(s

12449 – bask

- **12450** echo
- 12451 head

7	"/]	bin/e	echo h	i .	/usr,	/bin/l	head	d"],)	_
=	0									
=	1	<mark>2450</mark>	fd		bas	sh				
=	0		0		torm	inal				
=	1	2451	U		term	mai				
=	0		1		term	inal				
=	0									
			2		term	inal				
=	1		3		nine	read				
=	0				pipc_	rcau				
=	0		4		pipe_	write				
=	0									
n/	hea	ad"],)	= 0						
П	hi	"], .) =	0						
)	& &	WEXI	TSTAT	US (S) ==	0}],	0,	NULL) =	124
)	& &	WEXI	TSTAT	US (S) ==	0}],	0,	NULL) =	124









<pre>execve("/bin/bash", ["bash", "-c"</pre>
pipe([3, 4])
clone()
close(4)
clone()
close(3)
close(3)
<pre>wait4(-1, <unfinished></unfinished></pre>
dup2(4, 1)
dup2(3, 0)
close(3)
close(4)
<pre>execve("/usr/bin/head", ["/usr/bit</pre>
<pre>execve("/bin/echo", ["/bin/echo",</pre>
< wait4 resumed> [{WIFEXITED(s
<pre>wait4(-1, <unfinished></unfinished></pre>
< wait4 resumed> [{WIFEXITED(s

12449 – basł

- **12450** echo
- 12451 head

,	" /]	oin/e	cho h	i ,	/usr,	/bin/	/head	d"], .)	=
=	0									
=	12	2450	fc		bas	sh	ba	sh'		
=	0	2451	0		term	inal	tern	ninal		
=	0		1		term	inal	tern	ninal		
			2		term	inal	tern	ninal		
=	1		3		pipe_	read	pipe_	_read		
=	0		4		pipe_	write	pipe_	write		
= n/1	0 hea	ad"],)	= 0						
	hi	"], •) =	0						
)	& &	WEXI	TSTAT	US (S)) ==	0}],	, 0,	NULL)	=	124
)	& &	WEXI	TSTAT	US(S)) ==	0}],	, 0,	NULL)	=	124









12449	<pre>execve("/bin/bash", ["bash", "-c"</pre>
12449	pipe([3, 4])
12449	clone()
12449	close(4)
12449	clone()
12449	close(3)
12450	close(3)
12449	<pre>wait4(-1, <unfinished></unfinished></pre>
12450	dup2(4, 1)
12451	dup2(3, 0)
12451	close(3)
12450	close(4)
12451	<pre>execve("/usr/bin/head", ["/usr/big</pre>
12450	<pre>execve("/bin/echo", ["/bin/echo",</pre>
12449	< wait4 resumed> [{WIFEXITED(s
12449	<pre>wait4(-1, <unfinished></unfinished></pre>
12449	< wait4 resumed> [{WIFEXITED(s

12449 – basł

- **12450 echo**
- 12451 head

7	"	/}	oin/echo	hi	/usi	:/bin	/head	d"], .)	=
=	=	0								
=	=	12	2450	fd	ba	ash	ba	sh'		
-	-	0		0	terr	ninal	tern	ninal		
			24 3 1		ton					
	=	0		1	terr	ninal	tern	ninal		
				2	terr	ninal	tern	ninal		
:	=	1		3	pipe	read	pipe	read		
	_	0			nino	writo	nino			
-	=	0		4	hihe	_wne	hihe ⁻	_write		
n,	/h	ea	ad"],	.) = (C					
I	"h	i'	"],)	= 0						
)	&	. &	WEXITST	ATUS (S	5) ==	= 0}]	, 0,	NULL)	=	124
)	&	. &	WEXITST	ATUS (S	5) ==	= 0}]	, 0,	NULL)	=	124









12449	<pre>execve("/bin/bash", ["bash", "-c"</pre>
12449	pipe([3, 4])
12449	clone()
12449	close(4)
12449	clone()
12449	close(3)
12450	close(3)
12449	<pre>wait4(-1, <unfinished></unfinished></pre>
12450	dup2(4, 1)
12451	dup2(3, 0)
12451	close(3)
12450	close(4)
12451	<pre>execve("/usr/bin/head", ["/usr/big</pre>
12450	<pre>execve("/bin/echo", ["/bin/echo",</pre>
12449	< wait4 resumed> [{WIFEXITED(s
12449	<pre>wait4(-1, <unfinished></unfinished></pre>
12449	< wait4 resumed> [{WIFEXITED(s

12449 – basł

- **12450** echo
- 12451 head

		hin/	echo h	i /	'usr	/hin/	/head	4″1		=
, =	= 0	×11/ (ubr/		neuc	*]/	•••)	
=	= 1	2450	fd		bas	sh	ba	sh'	ba	sh''
=	= 0 = 1	2451	0		term	inal	tern	ninal	terr	mina
=	= 0		1		term	inal	tern	ninal	terr	mina
	- 0		2		term	inal	tern	ninal	terr	mina
	= 1 = 0		3	Ŗ	oipe_	read	pipe_	_read	pipe	e_rea
=	= 0		4				pipe_	write		
= 1	= 0 /he	ad"1)	= 0						
1	'hi	"],) =	0						
)	& &	WEX	ITSTATU	JS(S)	==	0}]	, 0,	NULL) =	124
)	& &	WEX	ITSTAT	JS(S)	==	0}]	, 0,	NULL) =	124









12449	<pre>execve("/bin/bash", ["bash", "-c"</pre>
12449	pipe([3, 4])
12449	clone()
12449	close(4)
12449	clone()
12449	close(3)
12450	close(3)
12449	<pre>wait4(-1, <unfinished></unfinished></pre>
12450	dup2(4, 1)
12451	dup2(3, 0)
12451	close(3)
12450	close(4)
12451	<pre>execve("/usr/bin/head", ["/usr/big</pre>
12450	<pre>execve("/bin/echo", ["/bin/echo",</pre>
12449	< wait4 resumed> [{WIFEXITED(s
12449	<pre>wait4(-1, <unfinished></unfinished></pre>
12449	< wait4 resumed> [{WIFEXITED(s

12449 - bask

- **12450** echo
- 12451 head

7	",	/bin	l/e	cho h	i	/usr/	/bin/	'heac	d"],)	=
-	= (0									
=	= :	1245	0	fd		bas	sh	ba	sh'	ba	ish''
	= (= :	0 1245	1	0		term	inal	tern	ninal	terr	mina
=	= (0		1		term	inal	tern	ninal	terr	mina
	= (2		term	inal	tern	ninal	terr	mina
-	= :	1		3		pipe_	read	pipe_	_read	pipe	_rea
-	- (0		4				pipe_	_write		
:	= (/h	0 0	1		- 0						
. 1 /	"h:	i"],	•	· · ·) =	0						
)	& 8	& WE	XI	TSTAT	US (S) ==	0}],	0,	NULL) =	124
)	& 8	& WE	XI	TSTAT	US (S) ==	0}],	0,	NULL) =	124









12449	<pre>execve("/bin/bash", ["bash", "-c"</pre>
12449	pipe([3, 4])
12449	clone()
12449	close(4)
12449	clone()
12449	close(3)
12450	close(3)
12449	<pre>wait4(-1, <unfinished></unfinished></pre>
12450	dup2(4, 1)
12451	dup2(3, 0)
12451	close(3)
12450	close(4)
12451	<pre>execve("/usr/bin/head", ["/usr/big</pre>
12450	<pre>execve("/bin/echo", ["/bin/echo",</pre>
12449	< wait4 resumed> [{WIFEXITED(s
12449	<pre>wait4(-1, <unfinished></unfinished></pre>
12449	< wait4 resumed> [{WIFEXITED(s

12449 - bash

- **12450** echo
- 12451 head

7	"/]	bin/e	echo hi	_ /1	usr/	bin/	/head	1"], .)	=
=	= 0									
=	= 1	2450	fd		bas	h	ba	sh'	ba	ish''
=	= 0 = 1	<mark>2451</mark>	0	t	termi	nal	tern	ninal	terr	mina
=	= 0		1	1	termi	nal	tern	ninal	terr	mina
	- 0		2	1	termi	nal	tern	ninal	terr	mina
=	= 1		3				pipe_	<u>_read</u>	pipe	e_rea
=	= 0		4				pipe_	write		
= 1/	= 0 /he	ad"])	= 0						
'	'hi	"],) =	0						
)	& &	WEX	ITSTATU	JS(S)	==	0}]	, 0,	NULL)	=	124
)	& &	WEX	ITSTATU	JS(S)	==	0}]	, 0,	NULL)	=	124









12449	<pre>execve("/bin/bash", ["bash", "-c"</pre>
12449	pipe([3, 4])
12449	clone()
12449	close(4)
12449	clone()
12449	close(3)
12450	close(3)
12449	<pre>wait4(-1, <unfinished></unfinished></pre>
12450	dup2(4, 1)
12451	dup2(3, 0)
12451	close(3)
12450	close(4)
12451	<pre>execve("/usr/bin/head", ["/usr/big</pre>
12450	<pre>execve("/bin/echo", ["/bin/echo",</pre>
12449	< wait4 resumed> [{WIFEXITED(s
12449	<pre>wait4(-1, <unfinished></unfinished></pre>
12449	< wait4 resumed> [{WIFEXITED(s

12449 - ba	ash
------------	-----

- **12450** echo
- 12451 head

1	"/]	bin/e	echo hi	L /1	usr/	bin/	/head	1"], .)	=
=	= 0 = 12	2450	fd		bas	sh	ba	sh'	ba	sh''
=	= 0 = 1	2451	0	-	termi	nal	tern	ninal	terr	nina
=	= 0 = 0		1	1	termi	nal	tern	ninal	terr	nina
	- 0		2	-	termi	nal	tern	ninal	terr	nina
=	= 1 = 0		3					Ķ	oipe	_rea
=	= 0		4				pipe_	write		
= 1,	= 0 /hea	ad"],	•••)	= 0						
I	'hi	"], •) =	0						
)	& &	WEXI	.TSTATU	JS(S)	==	0}]	, 0,	NULL)	-	124
)	& &	WEXI	TSTATU	JS(S)	==	0}]	, 0,	NULL)	=	124









12449	<pre>execve("/bin/bash", ["bash", "-c"</pre>
12449	pipe([3, 4])
12449	clone()
12449	close(4)
12449	clone()
12449	close(3)
12450	close(3)
12449	<pre>wait4(-1, <unfinished></unfinished></pre>
12450	dup2(4, 1)
12451	dup2(3, 0)
12451	close(3)
12450	close(4)
12451	<pre>execve("/usr/bin/head", ["/usr/bin</pre>
12450	<pre>execve("/bin/echo", ["/bin/echo",</pre>
12449	< wait4 resumed> [{WIFEXITED(s
12449	<pre>wait4(-1, <unfinished></unfinished></pre>
12449	< wait4 resumed> [{WIFEXITED(s

12449 - bash

- **12450** echo
- 12451 head

7	"/]	bin/e	echo hi	_ /1	usr/	bin/	/heac	1"], .)	=
	= 0									
=	= 12	2450	fd		bas	sh	ba	sh'	ba	sh''
=	= 0 = 1	2451	0	-	termi	nal	term	ninal	terr	nina
=	= 0		1	i	termi	nal	pipe_	write	terr	nina
	- 0		2	-	termi	nal	term	ninal	terr	nina
=	= 1		3					r	oipe	_rea
=	= 0 = 0		4				pipe_	write		
- ת ה	/hea /hea	ad"], "1	· · · ·)	= 0						
)	& &	WEX]	TSTATU	JS(S)	==	0}]	, 0,	NULL)	=	124
)	& &	WEX]	TSTATU	JS(S)	==	0}]	, 0,	NULL)	=	124









12449	<pre>execve("/bin/bash", ["bash", "-c"</pre>
12449	pipe([3, 4])
12449	clone()
12449	close(4)
12449	clone()
12449	close(3)
12450	close(3)
12449	<pre>wait4(-1, <unfinished></unfinished></pre>
12450	dup2(4, 1)
12451	dup2(3, 0)
12451	close(3)
12450	close(4)
12451	<pre>execve("/usr/bin/head", ["/usr/bit</pre>
12450	<pre>execve("/bin/echo", ["/bin/echo",</pre>
12449	< wait4 resumed> [{WIFEXITED(s
12449	<pre>wait4(-1, <unfinished></unfinished></pre>
12449	< wait4 resumed> [{WIFEXITED(s

12449	—	bash
-------	---	------

- **12450** echo
- 12451 head

7	"/]	oin/e	echo hi	. /1	usr/	/bin/	/head	1"],)	=
	= 0									
=	= 12	<mark>2450</mark>	fd		bas	sh	ba	sh'	ba	ish''
=	= 0 = 12	2451	0	1	termi	nal	term	ninal	pipe	e_rea
=	= 0		1	1	termi	nal	pipe_	write	terr	mina
	= U		2	i	termi	nal	term	ninal	terr	mina
=	= 1		3						pipe	e_rea
=	= 0 = 0		4				pipe_	write		
ר. י	/hea /hea	ad"], "1.	· · ·) =	= 0 0						
)	& &	WEXI	TSTATU	S(S)	==	0}]	, 0,	NULL) =	124
)	& &	WEXI	TSTATU	S(S)	==	0}],	, 0,	NULL) =	124









12449	<pre>execve("/bin/bash", ["bash", "-c"</pre>
12449	pipe([3, 4])
12449	clone()
12449	close(4)
12449	clone()
12449	close(3)
12450	close(3)
12449	<pre>wait4(-1, <unfinished></unfinished></pre>
12450	dup2(4, 1)
12451	dup2(3, 0)
12451	close(3)
12450	close(4)
12451	<pre>execve("/usr/bin/head", ["/usr/bit</pre>
12450	<pre>execve("/bin/echo", ["/bin/echo",</pre>
12449	< wait4 resumed> [{WIFEXITED(s
12449	<pre>wait4(-1, <unfinished></unfinished></pre>
12449	< wait4 resumed> [{WIFEXITED(s

- **12450** echo
- 12451 head

1	"/]	bin/e	echo hi	L /1	usr/	bin/	/heac	1"],)	=
=	= 0 = 1	2450	fd		bas	sh	ba	sh'	ba	ish''
=	= 0 = 1	<mark>2451</mark>	0	1	termi	inal	term	ninal	pipe	e_rea
=	= 0 = 0		1	1	termi	inal	pipe_	write	teri	mina
	- 0		2	1	termi	nal	term	ninal	teri	mina
-	= 1 = 0		3						pipe	e <u>rea</u>
=	= 0		4				pipe_	write		
= 1/	= 0 /hea	ad"],	••••)	= 0						
I	'hi	"],) =	0						
)	8	WEX]	TSTATI	JS(S)	==	0}]	, 0,	NULL) =	124
)	& &	WEXI	TSTATU	JS(S)	==	0}]	, 0,	NULL) =	124









12449	<pre>execve("/bin/bash", ["bash", "-c"</pre>
12449	pipe([3, 4])
12449	clone()
12449	close(4)
12449	clone()
12449	close(3)
12450	close(3)
12449	<pre>wait4(-1, <unfinished></unfinished></pre>
12450	dup2(4, 1)
12451	dup2(3, 0)
12451	close(3)
12450	close(4)
12451	<pre>execve("/usr/bin/head", ["/usr/big</pre>
12450	<pre>execve("/bin/echo", ["/bin/echo",</pre>
12449	< wait4 resumed> [{WIFEXITED(s
12449	<pre>wait4(-1, <unfinished></unfinished></pre>
12449	< wait4 resumed> [{WIFEXITED(s

12449 - bash

- **12450** echo
- 12451 head

, II ,	/bin/e	echo hi	/usr	/bin	/heac	1"], .)	=
= (0 12450	fd	ba	ish	ba	sh'	ba	sh''
= (0 12451	0	tern	ninal	term	ninal j	oipe	_rea
= (0	1	tern	ninal	pipe_	write	terr	nina
		2	tern	ninal	term	ninal	terr	nina
= (1 0	3						
= (0	4			pipe_	write		
= (n/h) "h.	U ead"], i"],	· · ·) = 0	• 0					
) &	& WEX	TSTATUS	(s) ==	0}]	, 0,	NULL)	=	124
) &	& WEX	TSTATUS	(s) ==	0}]	, 0,	NULL)	=	124









12449	<pre>execve("/bin/bash", ["bash", "-c"</pre>
12449	pipe([3, 4])
12449	clone()
12449	close(4)
12449	clone()
12449	close(3)
12450	close(3)
12449	<pre>wait4(-1, <unfinished></unfinished></pre>
12450	dup2(4, 1)
12451	dup2(3, 0)
12451	close(3)
12450	close(4)
12451	<pre>execve("/usr/bin/head", ["/usr/bit</pre>
12450	<pre>execve("/bin/echo", ["/bin/echo",</pre>
12449	< wait4 resumed> [{WIFEXITED(s
12449	<pre>wait4(-1, <unfinished></unfinished></pre>
12449	< wait4 resumed> [{WIFEXITED(s

- **12450** echo
- 12451 head

, _	"/}	oin/e	echo hi	i /	usr/	bin/	/head	l"],)	=
	= 12	2 450	fd		bas	sh	ba	sh'	h	ad
	= 0 = 12	2451	0	-	termi	inal	term	ninal	pipe	e_rea
=	= 0 = 0		1		termi	inal	pipe_	write	terr	nina
			2	-	termi	inal	term	ninal	terr	nina
=	= 1 = 0		3							
=	= 0		4							
n/	/hea	ad"],	•••)	= 0						
)	'hi' &&	WEXI) = [TSTAT]	0 JS(S)	==	0}]	, 0,	NULL) =	124
	2.2	WEXT	יחציים	IS(c)		011	0	NIIT.T.) =	124
)		ل ۲۵ تک ۷۷	. TOTAT(ر ۲ م	, •,	ЦОПП	/ _	141









12449	<pre>execve("/bin/bash", ["bash", "-c"</pre>
12449	pipe([3, 4])
12449	clone()
12449	close(4)
12449	clone()
12449	close(3)
12450	close(3)
12449	<pre>wait4(-1, <unfinished></unfinished></pre>
12450	dup2(4, 1)
12451	dup2(3, 0)
12451	close(3)
12450	close(4)
12451	<pre>execve("/usr/bin/head", ["/usr/bit</pre>
12450	<pre>execve("/bin/echo", ["/bin/echo",</pre>
12449	< wait4 resumed> [{WIFEXITED(s
12449	<pre>wait4(-1, <unfinished></unfinished></pre>
12449	< wait4 resumed> [{WIFEXITED(s

1	24	.49	 bash	1

- **12450** echo
- 12451 head

7		/ k	oin/	echo	hi	/	usr	/bin	he	ad"]	, •)	=
=	=	0											
=	=	12	2450		fd		bas	sh	e	echo		he	ead
	=	0	2451		0	-	term	inal	te	rmina	u p	oipe	_rea
:	=	0			1	+	term	inal	pip	e_wri	te	terr	nina
-	=	0			2		term	inal	te	rmina	ι	terr	nina
-	=	1			3								
	=	0			1								
	=	0			4								
л,	/h "h	ea	ad"]	, ••	.) =	0							
)	n &	1 &	J, WEX	···) ITST	= U ATUS	(S)	==	0}1	, 0	, NU	LL)	=	124
)	&	&	WEX	ITST	ATUS	(S)	==	0}]	, 0	, NU	LL)	=	124









When creating two children with a pipe (e.g., \$ cmd1 | cmd2), the pipe is created by the parent process before the first fork and ultimately closes both ends of the pipe. Why doesn't one the children create the pipe?

- B. The pipes are reused so that running a second pipeline like \$ cmd3 if the children created the pipe
- C. It doesn't matter which of the three processes (bash and the two

A. File descriptors are inherited by children so bash creates the pipe before either child so the children can communicate via the file descriptors

cmd4 doesn't require creating a new pipe. This wouldn't work

children) creates the pipe because prior to the exec()s, all three are copies of the same program (bash) so creating the pipe in any one creates them in all three. Bash just happens to do it in the parent.

Signals

Signals are how the kernel communicates with user processes

mask to see if the process is willing to accept receipt of the signal

- either the signal is ignored by default; or
- If it is willing, then the action depends on the **disposition** of the signal If the process is ignoring the signal, it's dropped If the process has installed a signal handler, the handler is run If the process has done neither, then the default action is performed
- - the process is terminated (or a small handful of other things)

If the signal is masked, then the signal remains pending until it is unmasked

When the kernel wants to signal the process, it checks the processes signal

Signal delivery

- Signal delivery is deferred until the kernel next returns to the process At the completion of a system call
 - The next time the process is scheduled to run

- Some system calls can be interrupted, others cannot System calls like read(2) and write(2) can read/write less than requested when interrupted by a signal; return value reflects this Other calls may return -1 and set errno to EINTR to indicate it was
 - interrupted

Only one of each (standard) signal may be pending at a time

Common signals: signal(7)

- SIGINT
- SIGQUIT
- SIGILL
- SIGABRT
- SIGFPE
- SIGKILL
- SIGSEGV
- SIGPIPE
- SIGTERM
- SIGCHLD
- SIGSTOP
- SIGCONT

- Interrupt from keyboard (ctrl-C on the terminal) Quit from keyboard (ctrl-\ on the terminal)
- Illegal instruction
- Signal from abort() (or assert() which calls abort()) - Floating point exception; integer divide by 0 on some systems - Kill signal, cannot be handled or ignored
- Segmentation fault
- Write to pipe with no readers
- Termination signal
- Child stopped or terminated Suspend the process (ctrl-Z on the terminal) - Resume the process (fg or bg on terminal)

- **SIGWINCH** Terminal window resized

Similar sounding signals

- Interrupt from keyboard (ctrl-C on the terminal) SIGINT Quit from keyboard (ctrl-\ on the terminal) SIGQUIT SIGKILL — Kill signal, cannot be handled or ignored

 - - Termination signal
- Suspend the process (ctrl-Z on the terminal) SIGSTOP

SIGTERM

SIGSTOP is about job control, not about terminating processes

- **SIGINT** and **SIGUIT** should only come from the user typing at the terminal
- If one process wants to stop another, it should (typically) request the process terminate via **SIGTERM** and, if after a few seconds it hasn't, use **SIGKILL**

Consider the following sequence of events

- The process installs a signal handler for SIGINT
- The process masks (blocks) SIGINT
- The user presses ctrl-c twice
- The process unmasks (unblocks) SIGINT

Which of the following is correct?

- A. The handler never runs
- B. The handler runs the first time ctrl-c is pressed
- C. The handler runs both times ctrl-c is pressed
- D. The handler runs once after the signal is unmasked
- E. The handler runs twice after the signal is unmasked

33

Sending a signal

From the shell: kill(1) or killall(1) \$ kill -9 1234 # Send SIGKILL (signal 9) to PID 1234 \$ kill -1 # List all of the signals

- int kill(pid t pid, int sig);
 - Sends signal sig to process pid
- int raise(int sig);
 - Sends signal sig to the own process

• Different behavior depending on pid < 0, pid = 0, pid > 0, sig = 0, sig > 0

Ignoring a signal/setting default

typedef void (*sighandler t)(int);

- sighandler t signal(int signum, sighandler t handler); Use SIG IGN for handler to ignore the signal
 - Use SIG DFL for handler to use the default behavior
 - You can also pass a function (pointer) of type void handler (int) but this isn't portable

Setting a handler portably

Use sigaction(2)

- Takes a pointer to a struct that holds the old handler and flags
- received while its handler is running
- Read the man page!

Takes a const pointer to a struct that holds a new handler and flags It flags specify the behavior of interrupted system calls, what information is given to the signal handler, and whether the same signal can be

Masking signals

Signal masks can be manipulated with sigprocmask(2)

Handling a signal

To handle a signal and then continue running, The signal handler should be installed with sigaction(2)

To handle a fatal signal and then exit as a result, The signal handler should be installed with sigaction(2) After performing any cleanup actions, the signal disposition should be reset to the default and the signal reraised

- - int handler(int sig) {
 - signal(sig, SIG DFL); raise(sig);

// Clean up actions, beware of signal handler limitations

In-class exercise



https://checkoway.net/teaching/cs241/2019-fall/exercises/Lecture-28.html

Grab a laptop and a partner and try to get as much of that done as you can!